MASTER EN SCIENCES INFORMATIQUES

Conception assistée de formulaires multimédias pour le commerce électronique dans le monde du World Wide Web

Alberty, Pascal; Derenne, Jérôme

Award date:
1997

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

Conception assistée de formulaires multimédias pour le commerce électronique dans le monde du World Wide Web

ALBERTY Pascal et DERENNE Jérôme

Mémoire réalisé en vue de l'obtention du diplôme de « Maître en Informatique »

Promoteurs : F. BODART et J. VANDERDONCKT

Année académique 1996-1997

FACULTES UNIVERSITAIRES NOTRE-DAME DE LA PAIX

NAMUR

Conception assistée de formulaires multimédias pour le commerce électronique dans le monde du World Wide Web

ALBERTY Pascal et DERENNE Jérôme

Mémoire réalisé en vue de l'obtention du diplôme de « Maître en Informatique »

Promoteurs : F. BODART et J. VANDERDONCKT

Année académique 1996-1997

Résumé

De plus en plus d'applications liées au commerce électronique sont développées et en particulier dans le monde du World Wide Web. Il existe des méthodologies et outils pour la création de ces applications hypermédias basées sur un schéma conceptuel et composées de formulaires.

Notre mémoire se propose d'analyser les apports de ces méthodologies et outils pour deux classes de problèmes. La première spécifie des applications hypermédias composées de formulaires utilisant des types de données simples. Nous étendons celle-ci en une deuxième classe en y ajoutant des éléments d'interaction et multimédias. Pour étudier cette dernière classe, nous présentons une étude de cas dont la finalité est le choix et l'achat d'une paire de lunettes solaires par un client via le World Wide Web.

Enfin, nous proposons des éléments de réflexion concernant une éventuelle méthodologie pouvant résoudre de manière adéquate cette seconde classe de problèmes.

More and more electronic commerce applications are developed and particularly for the World Wide Web. Some methodologies and tools for creating these hypermedia applications based on conceptual schema and composed by forms exist.

Our thesis propose to analyse contributions from these methodologies and tools for two kind of problems. First one specify hypermedia applications composed by forms using simple data types. We extend this one in a second class by adding interaction and multimedia elements. To study this last class, we present a study case which propose to a client to choose and to buy sunglasses via World Wide Web.

Finally, we give some elements of reflection about a possibly methodology which could be able to resolve appropriately this second class of problems.

Remerciements

Nous voudrions remercier les différentes personnes qui nous ont aidés pour la préparation et la rédaction de ce mémoire. Merci à Messieurs Bodart et Vanderdonckt qui nous ont donné l'opportunité de le réaliser.

Remercions également Carole Sumler, Khan Zhu, Curt Powley, David Benjamin et Ernesto Brodersohn de l'Information Sciences Institute de Los Angeles où nous avons réalisé notre stage de maîtrise. Leur accueil, leur amitié et leur expérience partagés ont fait de ce stage une expérience professionnelle très riche. De même, nous voudrions remercier toute la famille Scheding pour leur hébergement et l'expérience familiale partagée tout au long de cette période.

Un tout grand merci à Frédérique, à Guy, ainsi qu'à nos parents respectifs pour leur soutien moral, leurs conseils et le temps passé à la correction de ce mémoire.

Table des matières

Introduction

« New distribution channels, products and services have been created in order to satisfy the needs of a growing, wealthy- and technically-skilled population. These channels are based on technologies such as on-line services and the Internet, but also CD-ROMs and interactive TV » [PIGNEUR97].

L'utilisation monde interactif, et en particulier du support Internet, est en effet en pleine croissance. De plus en plus d'applications interactives sont développées pour ce support ; une telle plate-forme en favorise d'ailleurs le développement. En outre, les domaines couverts par ces applications, comme la consultation d'informations ou le commerce électronique, sont également en pleine extension. Ces considérations nous ont conduits à déterminer le cadre de travail du présent mémoire : l'étude des applications interactives développées pour la plate-forme Internet, dans le domaine particulier du commerce électronique.

En ce qui concerne ce type d'applications, deux manquements majeurs sont constatés. Tout d'abord, on remarque un suivi limité, voire inexistant, des méthodologies de développement. Ensuite, l'ergonomie de ces applications interactives est peu développée. Nous trouvons pourtant intéressant de rendre l'application accessible à une plus large population : une interface est en effet, selon F. Bodart : « Un système utilisé par une personne ou un groupe de personnes pour réaliser une tâche accomplie par un ensemble de moyens informatiques. Son objectif est d'effectuer la tâche de manière précise, rapide, sans erreurs et sans efforts inutiles, c'est-à-dire sans action étrangère à la tâche elle-même » [BODART93b]. Dans ce mémoire, nous considérons le premier manquement, à savoir la limitation des méthodologies existantes, étant donné que le second, par son ampleur, fait l'objet d'études dédicacées ; nous ne négligeons cependant pas ce dernier dans notre étude.

Dans un premier chapitre, nous décrivons des techniques d'interaction pouvant être utilisées par des applications hypermédias interactives, dans le cadre particulier du World Wide Web. Nous étudions également différentes méthodologies et outils, dont le but est de faciliter la construction de ce type d'applications.

Le chapitre deux consiste en une évaluation des apports des diverses méthodologies et outils envers la résolution d'une première classe de problèmes : la construction d'applications hypermédias simples, basées sur un schéma conceptuel simple.

Cette classe de problèmes étant résolue à partir de techniques existantes, nous l'étendons en lui ajoutant des éléments multimédias et interactifs. Nous constatons ensuite que les méthodologies proposées ne peuvent être appliquées directement à la classe de problèmes étendue.

Afin de concrétiser ce style de problèmes et de déterminer les éléments nécessaires à une modélisation adaptée, nous élaborons, dans le chapitre quatre, l'étude d'un cas particulier : cette application permet à un client de choisir une paire de lunettes solaires via une interface graphique implémentée pour le World Wide Web.

Enfin, à l'aide des données étudiées dans les deux chapitres précédents, quelques pistes de réflexion mettent en évidence, dans un dernier chapitre, les différents éléments à travailler afin de développer une éventuelle méthodologie de conception d'applications appartenant à la classe étendue.

Le lecteur pourra encore trouver, en Annexe A, le lexique reprenant la traduction et/ou la définition des termes plus techniques ou abréviations utilisés dans ce document. Ceux-ci seront signalés, lors de leur première apparition dans le texte, par le signe *.

Chapitre 1.
Etat de l'art

Afin de mieux comprendre le cadre de ce mémoire, nous présentons ici un état de l'art se présentant en trois sections. La première section aborde différentes techniques d'interaction que l'on peut rencontrer dans le monde du World Wide Web ; la seconde s'occupe de donner un descriptif d'outils pouvant être utilisés comme générateurs ou éditeurs de pages ou formulaires HTML ; suit alors la dernière section présentant différentes méthodologies de développement d'applications hypermédias. Cet ensemble de descriptions nous servira de base dans les chapitres suivants quant à l'étude de nos classes de problèmes.

1.1 Techniques d'interaction

Outil de diffusion de textes et d'images, le World Wide Web* s'oriente de plus en plus vers le multimédia. Pour ce faire, les applications clientes doivent supporter divers formats de fichiers multimédias. Ces différents formats peuvent être affichés, visualisés ou auditionnés ; la plupart des navigateurs actuels le permettent déjà.

Mais le terme « multimédia » sous-entend aussi une interaction de la part de l'utilisateur. Ce dernier doit pouvoir interagir avec le programme client Web via des techniques d'interaction multimédia. L'interaction résultante peut s'établir dans deux sens : du serveur vers le client, ou inversement. Dans le premier cas, l'utilisateur se contente la plupart du temps de « regarder » les informations, il est réceptif aux informations transmises par le serveur. Dans le second, il devient acteur et prend des initiatives suite à l'analyse des informations proposées par le serveur.

Que nous propose-t-on a l'heure actuelle, quelles sont les évolutions possibles, que doit-on attendre du Web ? Telles sont les questions auxquelles ce paragraphe tente de répondre.

1.1.1 Interaction du serveur vers le client

1.1.1.1 Les butineurs et le protocole Mime*

Les butineurs (ou navigateurs) de première génération étaient prévus uniquement pour l'affichage du texte. L'affichage d'éléments comme des images se déroulait en deux temps : une fois le fichier rapatrié via le protocole HTTP*, le butineur pouvait analyser son entête Mime ; celle-ci permettait alors de définir la nature du contenu du fichier et d'appeler le programme externe approprié.

Les formats multimédias définis implicitement par le protocole Mime [RFC1521] [EVRARD96] étaient à l'époque les suivants :

Pour les images :

- image/gif* : image au format gif (initialement créé pour le réseau CompuServe)
- image/jpeg* : image au format jpeg
- image/xbm* : image au format XBitmap (plus connu dans le monde Unix)

Pour le son :

- son/basic : son digitalisé sur 8 bits à 22 Khz, (extension AU*, mieux connu dans le monde Unix).

Pour la vidéo :

- video/mpeg* : vidéo au format mpeg
- video/quicktime* : vidéo au format quicktime (de Apple)

Le lecteur pourra trouver en Annexe B la liste complète des types et sous-types Mime ainsi que les différents documents les définissant.

1.1.1.2 Les Plugins et l'évolution des butineurs*

Au cours de leur évolution [HANDLEY95], les butineurs ont permis l'affichage direct de ces formats de fichiers ; de nombreux autres formats ont pu également être représentés. Par exemple le navigateur Netscape[1] version 3.0 complète permet l'affichage des formats gif, jpeg, mais aussi l'audition des sons aux formats au, wav*, aiff* et midi*, la visualisation des vidéos aux formats quicktime et avi*, ainsi que l'exploration des mondes virtuels VRML*.

Mais cette évolution peut être dynamique pour l'utilisateur. En effet, la société Netscape a permis une « personnalisation » de son butineur à l'aide de Plugins. Par exemple, si une entreprise invente un nouveau format d'image, elle peut créer un Plugin qui, une fois diffusé et installé par l'utilisateur, permettra la visualisation de ce format via le navigateur.

A l'heure actuelle, plus d'un centaine de Plugins existent. Ils recouvrent la majorité des formats de fichiers connus. Citons entre autres Word, Wordperfect, Excel et ABC pour la bureautique, AutoCad pour la CAO*, Tiff*, EPS* et TGA* pour les images, RealAudio et LiveVideo pour le son et la vidéo, ou encore Shockwave, IconAuthor et Adobe Acrobat pour les systèmes auteurs multimédias.

Ces Plugins peuvent être classés selon deux catégories. Les premiers permettent au butineur d'afficher directement dans la fenêtre principale le format dont il se charge ; ce système est alors totalement transparent pour l'utilisateur. Les seconds, par contre, sont composés d'un petit module externe au butineur, qui est appelé lors de la rencontre du document. La visualisation, par exemple, se fait alors en dehors du butineur.

Dans certains cas, lorsqu'un navigateur rencontre un document de type inconnu dans une page HTML*, il propose le chargement du Plugin adapté.

---

[1] Netscape Corporation http://www.netscape.com

Le futur Tag* HTML nommé EMBED devrait standardiser l'appel à une ressource sur le Web. En précisant une ressource avec ce Tag, le butineur recevra celle-ci précédée d'une en-tête Mime. Il reste alors au navigateur à comparer cette en-tête avec ses tables et soit à trouver la méthode pour l'afficher, soit à proposer de sauver la ressource, ou encore à proposer le chargement du Plugin adapté (si l'URL* où se trouve celui-ci est spécifié dans le Tag).

Prenons un exemple pour une Applet[2] Java* :

```
<EMBED src="example/Animator.class" width=100 height=100>
            <param name="text" value="Hello World!">
</EMBED>
```

ou pour un fichier vidéo:

```
<EMBED src="video/Hello.avi" width=100 height=100>
</EMBED>
```

1.1.1.3 Java et les butineurs auto-adaptateurs

Cette évolution dynamique des navigateurs atteint son plus haut niveau à l'aide du langage de programmation Java[3]. En effet, celui-ci permet d'écrire une Applet qui joue le rôle du Plugin. A cette différence près que l'Applet est indépendante du butineur et de la plate-forme sur laquelle le programme s'exécute. Ceci évite au programmeur d'écrire autant de versions de son programme que de systèmes existants.

Ajoutons que l'utilisateur ne doit plus intervenir : si le navigateur rencontre un format inconnu (et s'il supporte Java), il peut émettre une requête au serveur afin d'importer le code Java lui permettant de visualiser ce nouveau format. Il s'adapte donc seul !

1.1.1.4 PUSH

Nouvelle technologie en pleine croissance, le « PUSH » [INSIDE97] est un terme générique qui couvre une multitude de technologies différentes, reposant sur le même paradigme : pousser (push) l'information sur l'écran plutôt que d'obliger l'utilisateur à l'y tirer (pull).

La technologie « PUSH » s'inspire d'une métaphore télévisuelle : l'utilisateur s'abonne une fois pour toutes à un canal d'informations qu'il désire visualiser, et le contenu de ce canal est diffusé vers lui périodiquement. Quand, au moyen de son navigateur, il clique sur un hyperlien, une requête (un pull) est adressée à un serveur HTTP qui transmet l'information voulue au logiciel client. Dans une approche « PUSH », le serveur n'attend pas que l'utilisateur réclame une page : il la lui envoie dès qu'elle est prête.

---

[2] cfr. définition infra 1.1.2.4

[3] Java est un produit Sun - Javasoft : http://javasoft.sun.com

Cette technologie, chère aux annonceurs publicitaires, donne l'initiative au serveur et limite l'interaction de l'utilisateur au choix des informations qu'il désire recevoir. Son succès actuel provient de cette limitation. L'utilisateur ne doit plus fouiller et trier l'énorme quantité d'informations contenue par le Web pour trouver la page pertinente. Il lui suffit d'indiquer une fois pour toutes ce qu'il désire et d'attendre que cette information lui soit envoyée par le serveur.

1.1.2 Interaction du client vers le serveur

Le multimédia autorise un certain niveau d'interaction avec l'utilisateur. La première liberté qui lui est proposée est de choisir sa destination de navigation en cliquant sur les différents liens hypertextes proposés par le document. Bien vite, d'autres moyens d'interactivité sont apparus ; nous en abordons quelques-uns ci-après en présentant pour chacun une fiche technique, suivie d'explications et d'exemples.

1.1.2.1 Images cliquables

Nom français: Image cliquable

Nom anglais: Clickable picture

Autres: Clickable map, Image map

Définition: Image dont certaines zones sont liées à une URL. Quand l'utilisateur clique sur l'une de ces zones, l'URL correspondante est ouverte.

Principe: Un éditeur d'images cliquables affiche l'image choisie et l'utilisateur peut y définir des zones (rectangulaires, polygonales, circulaires ou ovales). Il lie alors chacune de ces zones à une URL. Une fois l'image entièrement éditée, elle peut être intégrée dans une page HTML classique.

Explications et exemples: Une des interactivités proposées sur le Web, et une des plus répandues, est sans doute la navigation par sélection d'une zone sur une image. Qu'elle se fasse par le script CGI* « ISMAP », les Tags Netscape « MAP et USEMAP » ou par le nouveau Tag HTML 3.0 « FIG », le principe reste le même : pour une image affichée, des zones ont été sélectionnées et correspondent chacune à une URL. Une URL par défaut est prévue pour les zones non couvertes.

Les zones peuvent être rectangulaires, circulaires ou même polygonales ; des programmes dédiés permettent de les définir. De plus, les éditeurs de pages HTML proposent de plus en plus un module ayant cette fonction.

Voici, obtenu par une capture d'écran, un exemple de création d'une image cliquable réalisée avec Map This, programme freeware* :

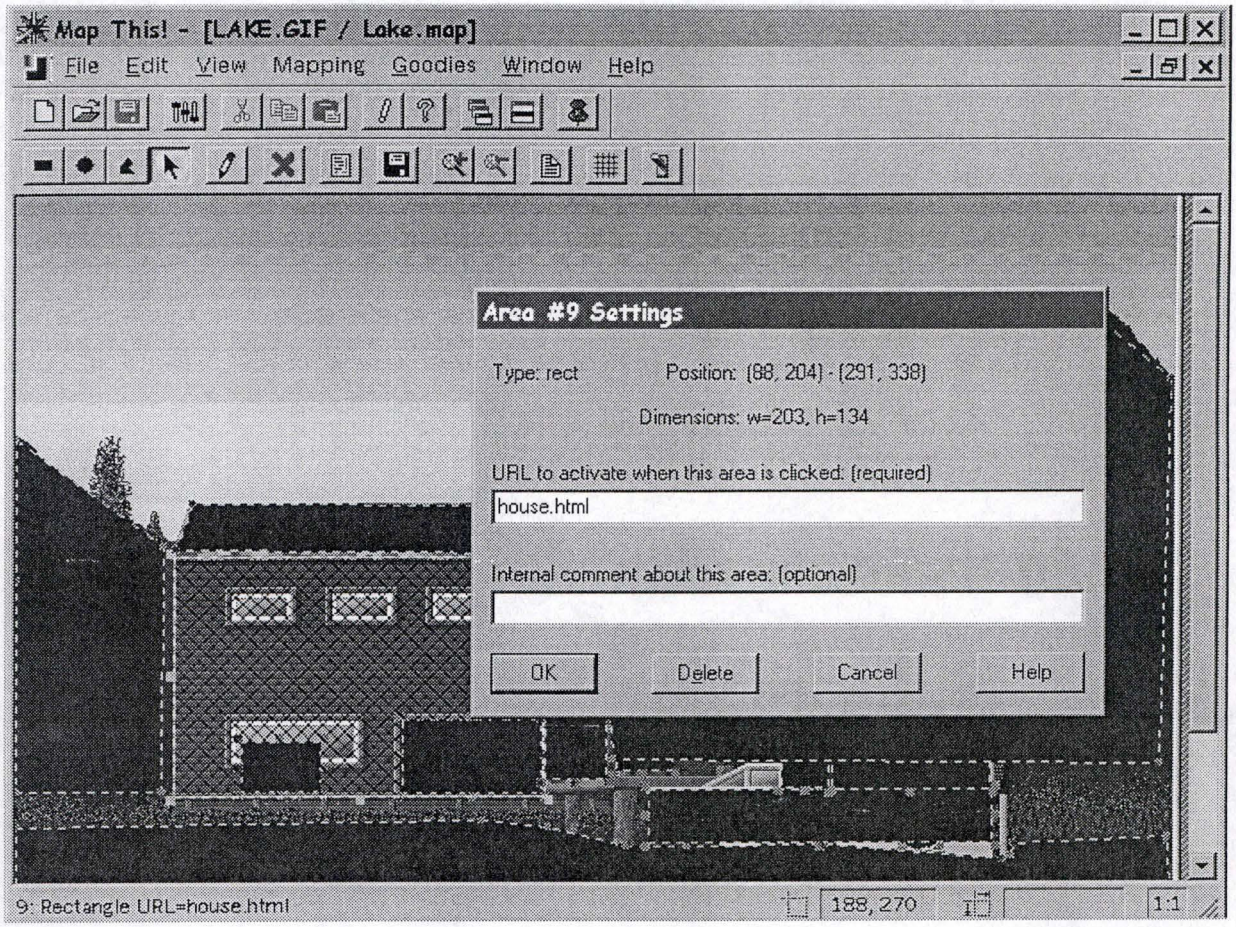


Figure 1-1 : Exemple d'édition d'image map avec MapThis

Exemple d'utilisation des Tags MAP (définition des zones sensibles) et USEMAP (attribut d'une image précisant le MAP à utiliser) :

```
<MAP NAME="buttonbar">
<AREA SHAPE="RECT" COORDS="10,10,49,49" HREF="about_us.html">
<AREA SHAPE="RECT" COORDS="60,10,99,49" HREF="products.html">
<AREA SHAPE="RECT" COORDS="110,10,149,49" HREF="index.html">
<AREA SHAPE="RECT" COORDS="0,0,159,59" NOHREF>
</MAP>
<IMG SRC="../images/tech/bar.gif" USEMAP="#buttonbar">
```

Une image peut être utilisée comme image cliquable aussi bien via CGI que via le mécanisme de Netscape, il suffit de préciser les deux attributs. Ce qui donne:

```
<A HREF="/cgi-bin/imagemap/pic2">
<IMG SRC="../images/pic2.gif" USEMAP="maps.html#map2" ISMAP></A>
```

1.1.2.2 Formulaires et HTTP File Upload

Nom français: Formulaire

Nom anglais: Form

Définition: Composition d'éléments d'interaction permettant la transmission d'informations structurées, aussi bien dans le sens Client-Serveur (question) que Serveur-Client (réponse).

Principe: Un formulaire fait partie d'une page HTML, il peut donc être édité en tant que tel. L'information véhiculée par le formulaire peut être prise en charge, par exemple, par un script CGI.

Explications et exemples: Les formulaires constituent un autre moyen de dialogue entre le Serveur Web et son utilisateur. Ce mode ouvre de nombreuses possibilités d'application. Les plus connues et répandues sont les recherches dans des catalogues (URL, bibliothèques, ...), l'inscription en tout genre (Mailing List*, concours, ...) et bien évidemment le commerce électronique (achats divers, recherche dans des catalogues de produits par exemple).

Les formulaires peuvent être composés des éléments suivants : champs d'édition, boutons radio, boîtes à cocher, listes de sélection (déroulantes ou non), boutons de commandes. Une fois le formulaire complété, les informations sont envoyées à une URL ou à une adresse électronique. Celles-ci peuvent être traitées par le serveur, par exemple via un script CGI. La réponse à la requête de l'utilisateur se compose bien souvent d'une nouvelle page HTML.

Le « HTTP File Upload », option intéressante proposée par Netscape, permet à un utilisateur de transmettre un fichier à une URL via le protocole HTTP. En voici un exemple :

```
<HTML>
<HEAD>
<TITLE>HTTP File Upload</TITLE>
</HEAD>
<BODY>
<FORM ENCTYPE="multipart/form-data" ACTION="_URL_" METHOD=POST>
Send : <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="Submit" VALUE="Send File">
</FORM>
</BODY>
</HTML>
```

Ce qui donne à l'écran :

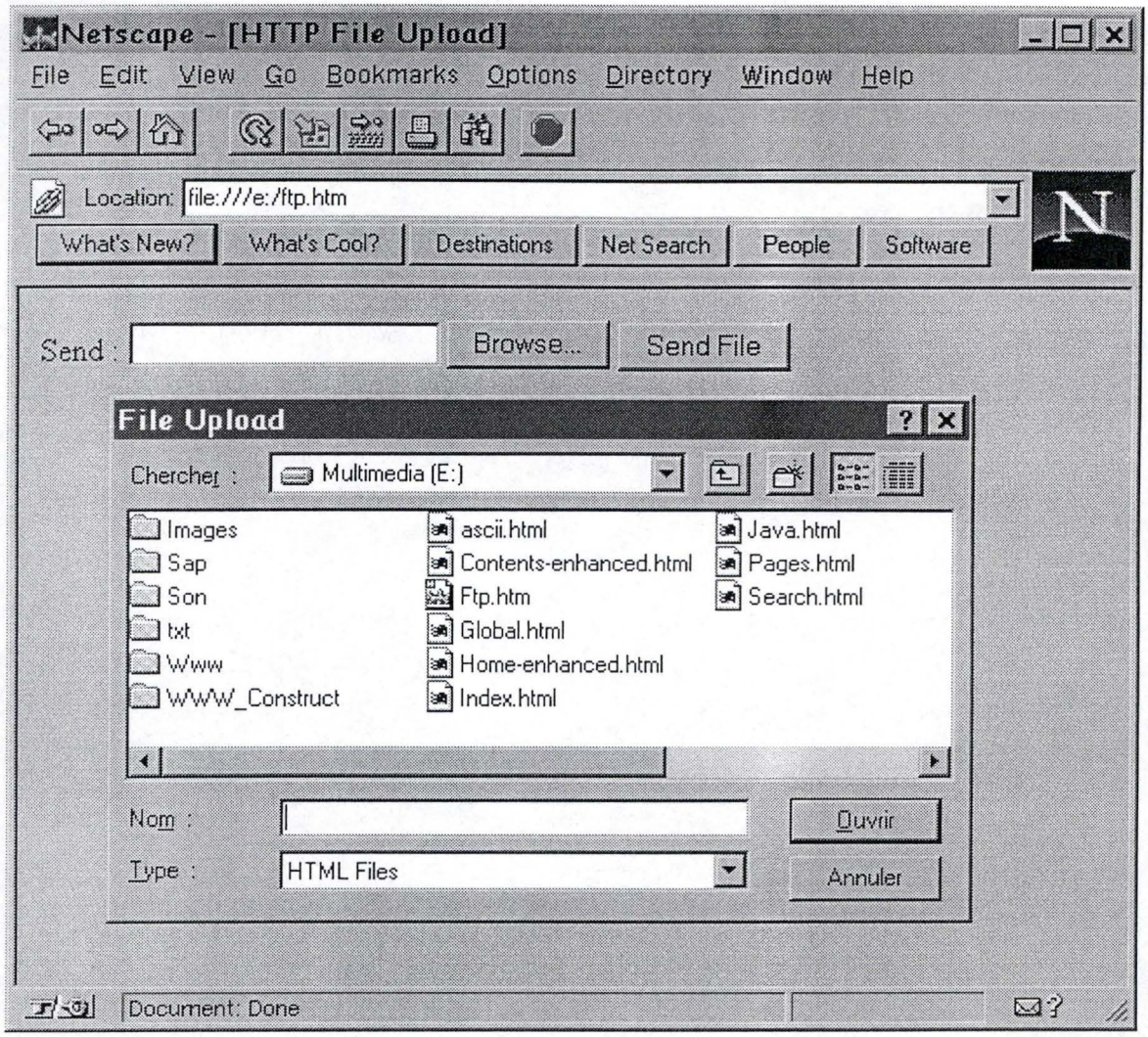


Figure 1-2 : Exemple de « HTTP File Upload » de Netscape

1.1.2.3 VRML[4]

Nom français: Langage de modélisation de réalités virtuelles.

Nom anglais: Virtual Reality Modelling Language.

Définition: Un fichier VRML reprend ce que l'utilisateur voit dans une scène de réalité virtuelle : il décrit les formes géométriques et leurs attributs visuels. Il comprend également les différentes possibilités d'interaction.

Principe: Comme indiqué dans la définition, pour créer un fichier VRML, il faut préciser les objets et les propriétés du monde élaboré. Certains programmes permettent de dessiner cet environnement et de construire un fichier VRML.

[4] VRML est un produit Silicon Graphics : http://vrml.sgi.com

Explications et exemples: Un fichier VRML décrit un monde virtuel en trois dimensions. A sa visualisation, le butineur permet alors une navigation en trois dimensions correspondant aux modes de marche d'un humain, de vol d'un avion, etc. Il est possible de se rapprocher d'un objet, de le contourner, de cliquer dessus. Une URL peut être associée à un objet. Le fait de cliquer sur celui-ci provoque alors la requête de chargement du document identifié par cette adresse.

Un programme comme Caligari Pioneer[5] permet l'édition de mondes virtuels en 3D et l'attribution d'URL à des objets de ce monde. Il permet en outre d'associer un son tridimensionnel à un objet. Lorsque l'on se déplace autour de l'objet, la sensation n'en est que plus réaliste !

1.1.2.4 JAVA

Nom: JAVA (nom propre)

Définition: Langage de programmation orienté objet créé par Sun et permettant, entre autres, la création d'Applets, petits programmes importables via le Web et s'exécutant sur la machine appelante, indépendamment du système de celle-ci.

Principe: Une application ou Applet est programmée et compilée grâce au compilateur du JDK* (appelé javac). Le « byte-code » (indépendant de toute plate-forme) peut être exécuté en 'stand-alone' (via l'interpréteur appelé java), ou via le Web et un navigateur compatible dans le cas d'une Applet.

Explications: Bien que Java ne puisse être considéré comme un format de fichier multimédia, nous nous permettons de le citer ici car ce langage permet, entre autres, d'écrire des applications multimédias hautement interactives via Internet. Java supporte la programmation d'applications reposant sur les protocoles TCP* ou UDP*. Il est donc possible d'écrire une application client FTP*, WWW, ou tout autre application originale utilisant son propre protocole.

Les Applets écrites avec Java s'insèrent directement dans une page Web et sont interprétées par la machine virtuelle comprise dans le butineur ; l'Applet est donc exécutée indépendamment de la plate-forme sur laquelle elle réside. Un même programme pourra donc être utilisé sous Unix, comme sous Windows ou Mac/OS, conformément à la philosophie d'Internet d'ouverture au monde.

[5] Pionner est un produit Caligari: http ://www.caligari.com/

Il existe de nombreux environnements de programmation Java. Ceux-ci reposent le plus souvent sur le JDK de Sun qui comprend dans sa version 1.0.2 le compilateur (javac), l'interpréteur (java), l'appletviewer (appletviewer), le debuggeur (javab), ainsi que quelques petits utilitaires.

IFC - Internet Fondation Classes (Netscape) : Les classes IFC de Netscape composent une librairie d'objets Java destinés à permettre aux programmeurs de software la construction et le déploiement rapide d'applications de nouvelle génération, les applications on-line via réseau.

Les objets contenus dans les IFC facilitent la création d'applications intégrant:

- des fenêtres,
- des contrôles à Interface-Utilisateur avancé tels que les 'color picker' ou le 'font chooser'[6],
- des animations,
- le 'drag and drop'*,
- des Timers,
- du texte multi-fontes.

1.1.2.5 ActiveX[7] (Microsoft)

Nom: ActiveX* (nom propre)

Définition: ActiveX est une collection de technologies qui permet aux composants d'un programme d'interagir avec un autre composant d'un autre programme et ce dans un environnement réseau, indépendamment du langage dans lequel ils ont été créés (définition Microsoft).

Principe: Les éléments ActiveX sont intégrés dans une page HTML et communiquent avec d'autres composants du même type ou d'un type différent.

Explications: De part leur modularité, les composants peuvent être intégrés à une page Web afin d'en augmenter l'interactivité. Un son, un logo animé, des boutons, des formulaires, beaucoup d'éléments peuvent être ajoutés. Ceux-ci peuvent être complexes ou même des morceaux de programme. Les éléments ActiveX sont directement concurrents au langage de programmation JAVA.

[6] Boîtes de dialogue prédéfinies ayant pour but le choix d'une couleur ou d'un type de caractère
[7] ActiveX est un produit Microsoft : http://www.microsoft.com

Cependant, Netscape, avec sa technologie Netscape Beans, essaie de standardiser cela en spécifiant un type d'élément "universel", capable de communiquer par exemple avec les éléments ActiveX, OCX*, DDE* et OpenDoc*.

1.1.3 Le cas Power Point: Internet Assistant et Plugin

Citons encore le cas de Power Point de Microsoft et de ses compléments, compris dans la série Internet Assistant d'Office. Power Point constitue en effet une référence en matière de présentation multimédia.

Deux compléments existent pour ce logiciel. Le premier est un Plugin utilisé pour la visualisation des fichiers au format Power Point (extensions ppt* ou ppz*). La présentation des transparents Power Point se déroule alors dans la fenêtre du navigateur, exactement comme dans le programme original. Un exemple est montré par la figure suivante :

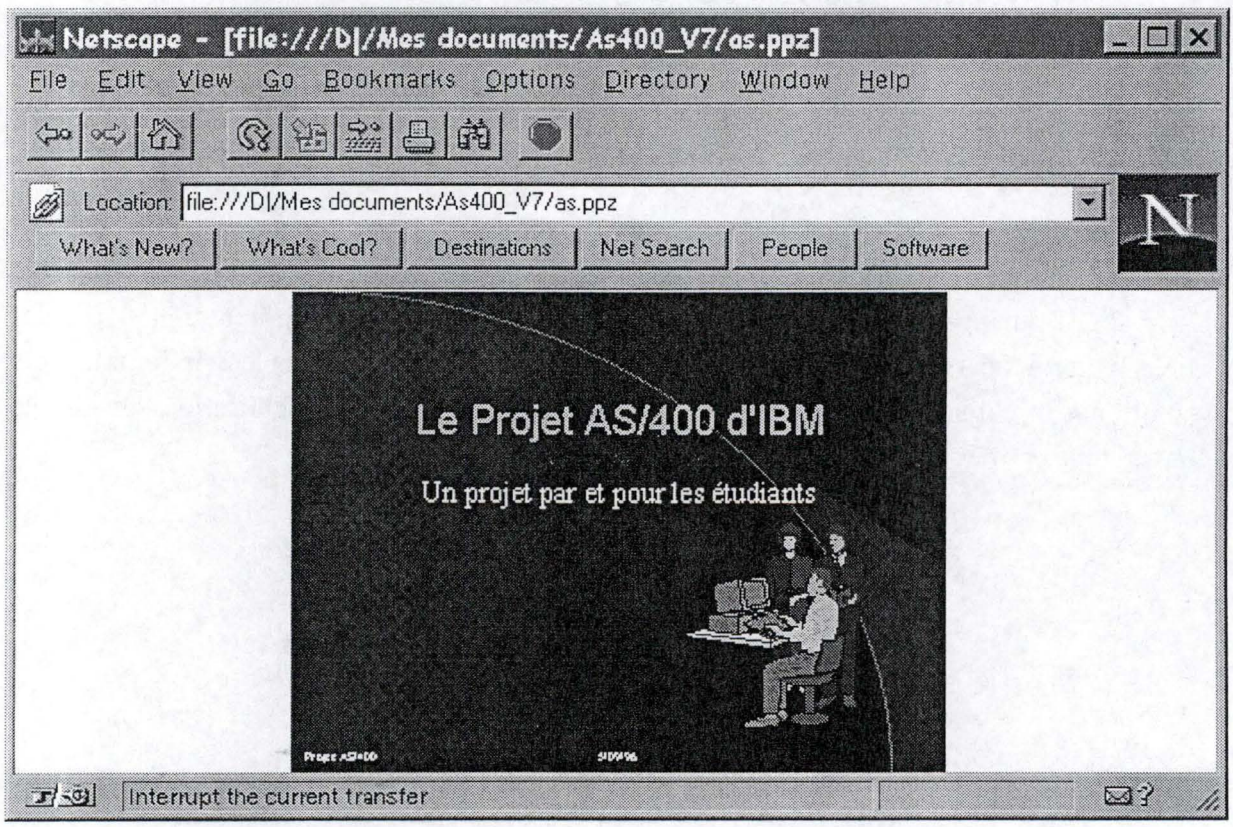


Figure 1-3 : Visualisation d'un fichier PowerPoint dans un navigateur à l'aide du Plugin

Le second complément, l'Internet Assistant, permet l'exportation des présentations sous deux formats. Le premier propose une page HTML dans laquelle l'objet a été inséré ; la page et la présentation doivent alors être visionnées à l'aide du Plugin précité.

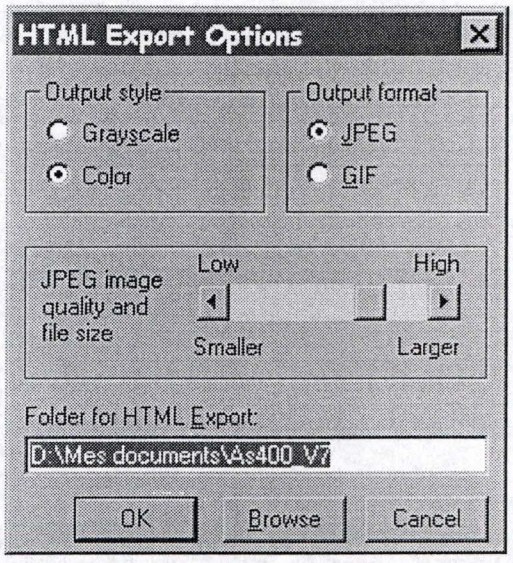


Figure 1-4 : Options du Plugin Internet Assistant pour PowerPoint

La seconde solution consiste à produire deux pages HTML par transparent Power Point, une en version texte, l'autre en version graphique (les diapositives sont converties en format Gif ou Jpg). Les différentes pages de même version sont reliées entre elles par des liens hypermédias (textes dans le premier cas, boutons graphiques dans le deuxième). Cette solution n'implique donc pas l'obligation de posséder le Plugin de visualisation. Un simple navigateur permet de proposer la séries de transparents à l'utilisateur.

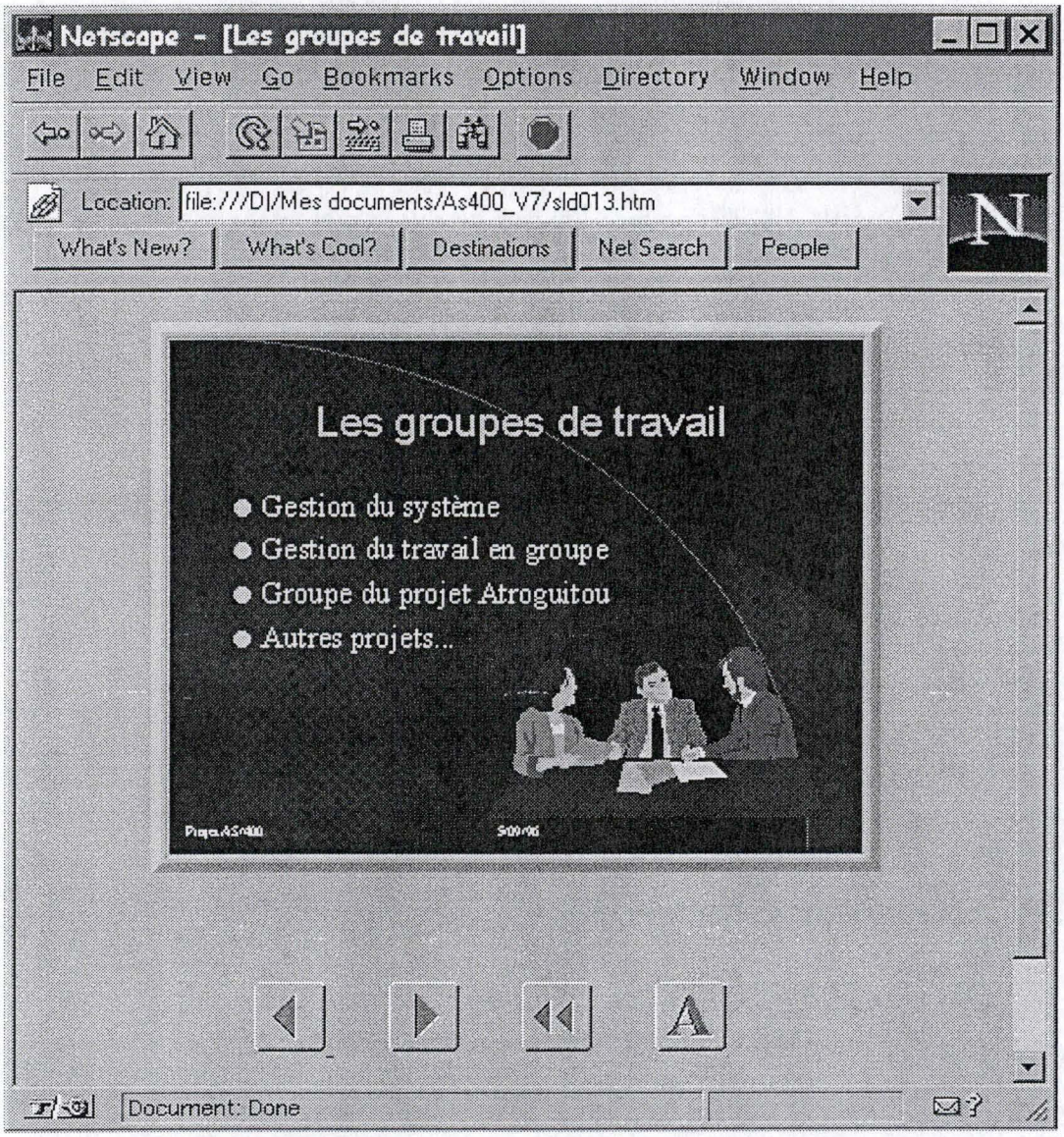


Figure 1-5 : Visualisation des transparents exportés à partir de PowerPoint à l'aide de l'Internet Assistant

On peut remarquer sur la Figure 1-5 que la page visualisée est un fichier au format HTML, composé d'une image GIF représentant le transparent et d'un ensemble de boutons graphiques servant à la navigation entre pages. Au contraire, dans la figure représentant la visualisation d'un fichier Power Point dans un navigateur à l'aide du Plugin (Figure 1-3), on peut repérer que l'URL utilisée spécifie un fichier au format Power Point.

Pour l'instant, ces deux compléments ne fonctionnent que sous Windows95 ou Windows NT. Notons également que depuis la version 97 de PowerPoint, l'Internet Assistant est inclus « de série ». Dans le même ordre d'idées, la suite bureautique « Office 97 » s'est ouverte à Internet.

1.2 Outils de conception assistée de pages HTML

La génération de pages HTML peut se réaliser de différentes manières et ce à l'aide de plusieurs types d'outils. Deux catégories d'outils de conception assistée se dégagent.

Les premiers sont des programmes « classiques » d'édition et de génération de pages HTML, à savoir des éditeurs de textes améliorés pour le domaine du Web. Comme exemples représentatifs de cette classe, nous étudions FrontPage 97[8] et WebMania[9].

Les seconds sont des programmes déjà plus évolués qui n'ont pas comme but premier de produire du code HTML mais plutôt soit de créer des applications interactives complètes, comme Designer 2000[10], soit de gérer une base de données orientée objet, comme EKTOS[11]. Cependant, ceux-ci sont cités car ils contiennent une option qui permet la génération escomptée. Ils sont présentés dans la seconde sous-section.

Enfin, le lecteur peut retrouver en Annexe C une étude et une comparaison d'autres outils d'édition de pages HTML et une liste exhaustive d'outils d'intégration de base de données.

1.2.1 Editeurs de pages HTML

1.2.1.1 FrontPage 97 - Microsoft Corporation

Aperçu. FrontPage 97 est implémenté en un programme indépendant pour la plate-forme MS-Windows 95/NT. Son affichage est de type WYSIWYG* et il comprend la vérification de la syntaxe HTML durant l'édition.

Environnement de développement de sites Web complets. Le programme FrontPage 97 contient tout ce dont l'utilisateur a besoin pour l'organisation et le développement d'un site Web. Le Personal Web Server est inclus, ainsi qu'un certain nombre d'extensions s'apparentant au CGI : recherche de documents, création de groupes de discussion, etc. Plusieurs options sont proposées, comme la création de formulaires et le traitement des informations liées à ces formulaires ; elles sont facilement réalisables par l'insertion d'un « bot » (bot étant un terme utilisé par Microsoft pour nommer les utilitaires orientés serveur).

---

[8] FrontPage 97 est un produit Microsoft : http://www.microsoft.com
[9] Webmania est un produit Q-D Software Development : http://www.q-d.com
[10] Designer 2000 est un produit Oracle : http://www.oracle.com
[11] EKTOS est un produit EKTELIS : http ://www.ektelis.be

Les bots permettent aussi de créer des librairies de code HTML pouvant être insérées dans les pages par le serveur avant l'envoi au butineur (ceci peut s'avérer pratique pour l'insertion d'une barre de navigation sur toutes les pages, par exemple).

Tableaux et édition de formulaires. Le programme permet à l'utilisateur de manipuler les formulaires et les tableaux graphiquement au sein de la fenêtre d'édition. Le bouton droit de la souris fait apparaître un menu « pop-up » autorisant l'utilisateur à modifier les attributs des éléments du formulaire.

Les assistants. Le nombre d'assistants est assez impressionnant. Avec l'un d'entre eux, il est possible de créer un site Web complet incluant les pages de recherches, les formulaires et la table des matières. D'autres assistants permettent quant à eux de créer un site de discussion. Comme pour tous les assistants Microsoft, toutes les étapes sont effectuées une par une, incluant un retour visuel des résultats potentiels.

Le support des Frames et de certaines extensions. Le programme FrontPage 97 contient un assistant, pour créer rapidement et graphiquement des formulaires. Malheureusement, il est impossible d'avoir un accès direct aux Tags des Frames une fois que la page est créée et l'utilisateur est donc obligé de recommencer l'opération (avec l'assistant) pour effectuer les changements désirés. Cet assistant utilise le pourcentage des Frames par rapport à la page afin de leur donner une dimension, ce qui est pénalisant si l'on désire choisir un nombre fixe de pixels (pour une en-tête graphique par exemple). Les attributs d'alignement et certaines autres spécifications d'HTML 3.0 sont permis, mais les Tags DIV, FIGS et ceux fournissant une aide à l'élaboration de formules mathématiques sont absents.

Frontpage 97 propose le support d'un plus grand éventail de Tags HTML (tels que les sons en arrière plan ou l'insertion de fichier vidéo AVI, propres à Microsoft Explorer), le support des contrôles ActiveX, des Applets Java, des Plugins Netscape, des animations PowerPoint, des scripts JavaScript ou de VBScript.

Les images cliquables (images maps) et les images transparentes. Le programme permet à l'utilisateur de définir les zones sensibles de son image au sein de l'éditeur. L'image cliquable correspondante est créée automatiquement par FrontPage 97. Il est également permis de spécifier une couleur « transparente »[12], simplement en sélectionnant un pixel de la couleur appropriée sur l'image.

Inconvénients de FrontPage. FrontPage Explorer est encore employé par l'utilisateur pour repérer le contenu de son site Web le plus facilement possible. Il contient des boîtes de dialogue optimisées notamment pour la création de liens entre pages d'un même serveur Web, défini par Explorer.

[12] Les pixels ayant cette caractéristique prendront comme attribut la couleur de fond de la page.

Malheureusement, en ce qui concerne les liens externes, il faut introduire les URL au clavier et ceci même pour les pages se trouvant sur un serveur local (et ne faisant pas partie du Web).

1.2.1.2 WebMania! 1.2 - Q&D Software Development

Le programme WebMania est implémenté en un programme indépendant pour la plate-forme MS-Windows 3.1/95/NT. Il est orienté texte et ne possède pas de vérification de la syntaxe HTML.

En voici une copie d'écran :

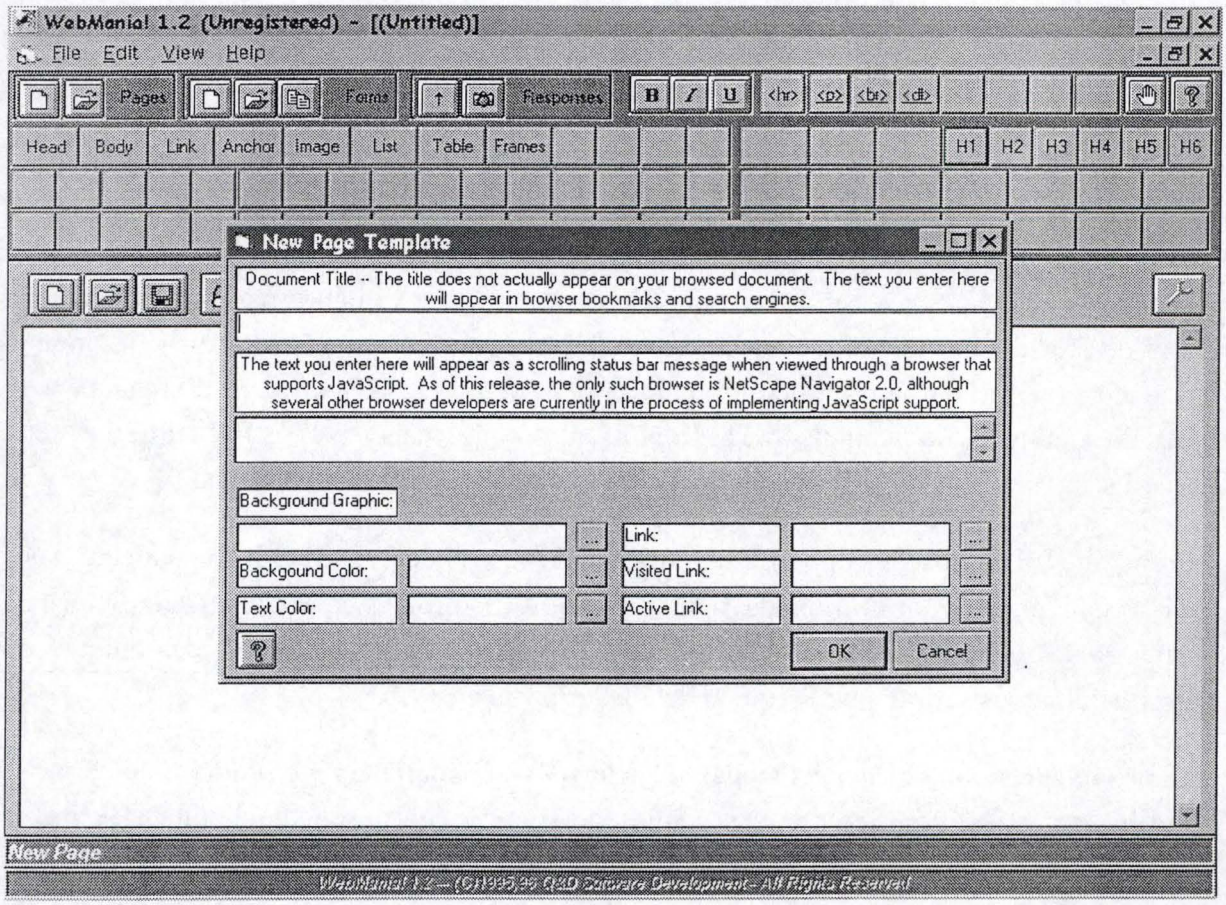


Figure 1-6 : Capture d'écran de WebMania

Aide à la gestion de formulaires. Ce programme contient, pour la création de formulaires, un lot d'assistants assez impressionnant. Pour l'utilisateur novice, cette aide peut sembler au départ un peu trop riche tant elle regroupe de notions, mais une fois l'apprentissage de l'outil accompli, il est facile de créer des formulaires et de réutiliser les éléments entre eux.

WebMania comprend aussi un outil permettant de lire les réponses envoyées par courrier électronique ; les formulaires standards créés avec WebMania incorporent par défaut le postage de la feuille par ce système. Ceci est pratique pour les utilisateurs qui n'ont pas accès au CGI sur leur système et qui ont besoin d'informations structurées quant au contenu de la réponse envoyée par la personne remplissant le formulaire.

Editeur de Frames et de tableaux. Un éditeur de Frames et de tableaux est également proposé afin de simplifier la gestion de ces éléments. De plus, beaucoup de boutons sont uniquement destinés à l'ajout de Tags personnalisés. L'utilisateur trouvera là un confort et une convivialité non négligeables.

Inconvénients de WebMania. Remarquons tout d'abord que les fonctions d'édition sont réduites. En effet, bien que l'on puisse ajouter ses propres Tags, WebMania en comprend peu de standards (par exemple la plupart des Tags concernant le formatage des paragraphes sont absents, ainsi que les Tags HTML 3.0). De plus, la gestion des liens au sein d'un document est minimale. Le programme semble avoir pour principale cible la gestion des formulaires ; l'organisation des tableaux et le support de JavaScript sont toutefois relativement bien réalisés.

De même, au point de vue ergonomie logiciel, on pourrait reprocher à WebMania de présenter à l'utilisateur des écrans surchargés d'informations parfois inutiles.

1.2.2 Autres outils

1.2.2.1 Designer 2000

Designer 2000 [GWYER96] [BARNES96] est un outil de software engineering développé par Oracle et permettant de créer des applications indépendantes ou du code HTML. Dans ce dernier cas, il existe deux façons d'obtenir une application complète en HTML. La première, classique, consiste à créer de toutes pièces les informations nécessaires ; la seconde emploie le module de « reverse engineering » de Designer 2000. Il est donc possible de générer une solution Web à partir d'une ancienne application développée avec ce programme.

Aperçu du Oracle WebServer. L'Oracle WebServer est intégré dans la base de données Oracle 7. Les données stockées dans celle-ci sont formatées dynamiquement en documents HTML par le WebServer et ensuite transmises au client Web à la demande de l'utilisateur (Figure 1-7).

L'utilisateur Web introduit une requête d'informations HTML en entrant une URL appropriée. Celle-ci spécifie une machine ou un site cible sur lequel s'exécute l'Oracle WebServer : elle précise également la session WebServer (càd les informations pour l'identification de l'utilisateur de la base de données) ainsi que la routine PL/SQL[13] à exécuter.

Cette routine spécifiée dans l'URL récupère les données dans la base Oracle et utilise d'autres routines PL/SQL afin de fournir les données formatées à l'utilisateur dans un format HTML adéquat. Ces informations sont envoyées à celui-ci via le Web Agent.

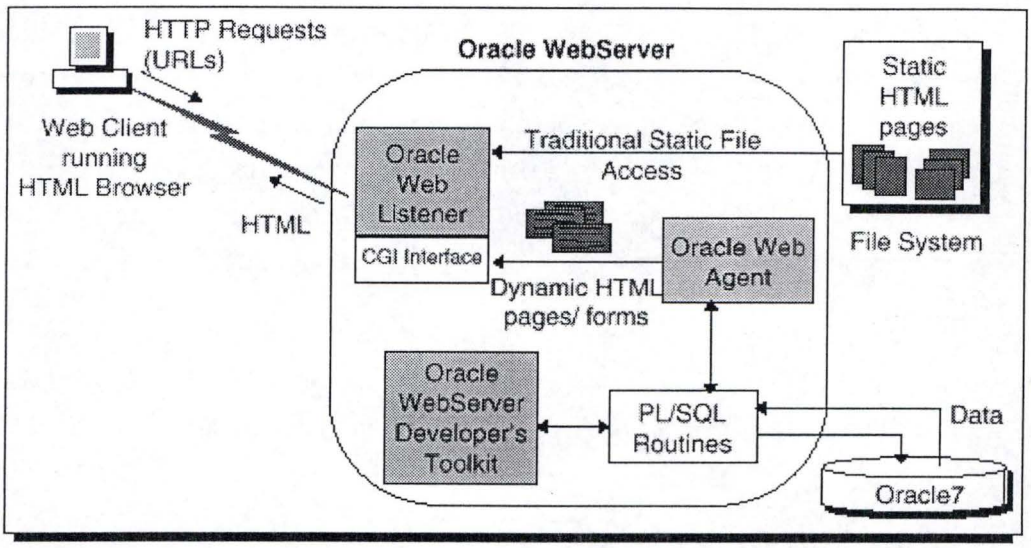


Figure 1-7 : Fonctionnement général de Oracle WebServer

Designer/2000 WebServer Generator. L'« Oracle Designer/2000 WebServer Generator » permet de construire des applications Web en utilisant les mêmes techniques de modélisation que les générateurs classiques de Designer 2000 (il permet en effet aussi de générer des applications Designer 2000 et Visual Basic) ; il se base sur les données contenues dans le repository du software. Ceci offre la possibilité de générer des solutions Web à partir de définition d'applications déjà construites en Designer 2000 ou Visual basic.

En utilisant Designer 2000, on peut présenter des pages statiques mais également des pages de requêtes interactives où un utilisateur peut entrer des paramètres de requête afin d'interroger la base de données. Cette requête questionne en temps réel la base de données Oracle 7 et le résultat au format HTML est présenté directement à l'utilisateur.

[13] Version étendue du langage de requête SQL propre aux bases de données Oracle

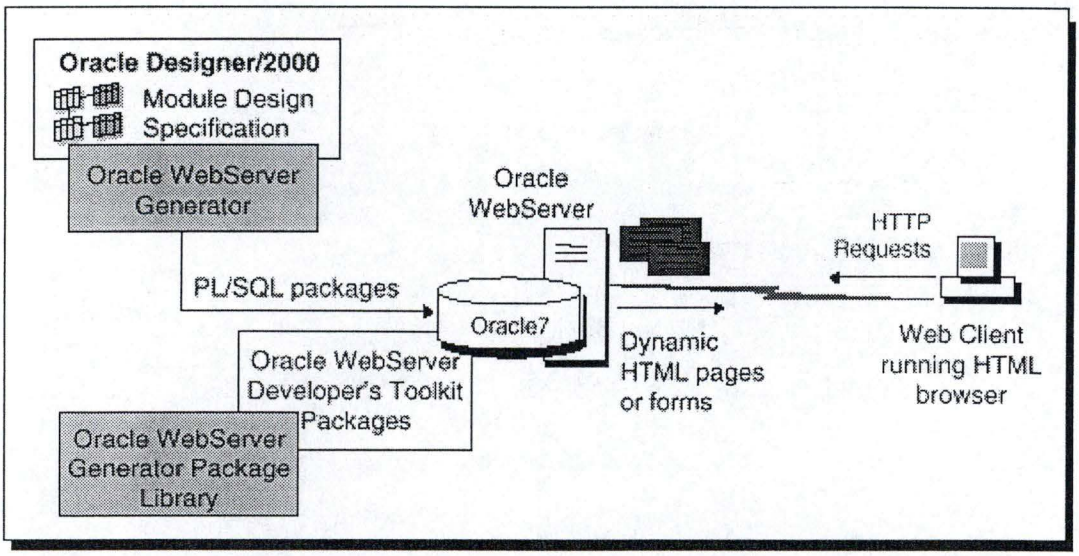


Figure 1-8 : Génération de pages HTML avec Oracle WebServer

La principale information requise par le processus de génération est la spécification d'un module de design et est contenue dans le composant Module Data Diagrammer de Designer 2000. Cette spécification contient les tables et colonnes utilisées par le module, les liens entre eux et l'information détaillée sur la manière d'utiliser les données.

Les modules peuvent être liés entre eux en utilisant le « Module Structure Diagrammer ». Dans l'application générée, ces liens permettent une navigation entre les modules via des liens hypertextes.

Durant la génération, le WebServer Generator crée un ensemble de routines PL/SQL qui sont installées dans la base de données du WebServer. Les préférences déterminent l'apparence générale de l'application générée et peuvent être configurées par l'utilisateur selon ses besoins.

L'Oracle PowerBrowser, ou tout autre navigateur HTML, peut utiliser les applications générées par l'appel de l'Oracle Web Agent qui construit dynamiquement les documents HTML à partir des scripts PL/SQL générés par le WebServer Generator.

Fonctionnement du WebServer Generator

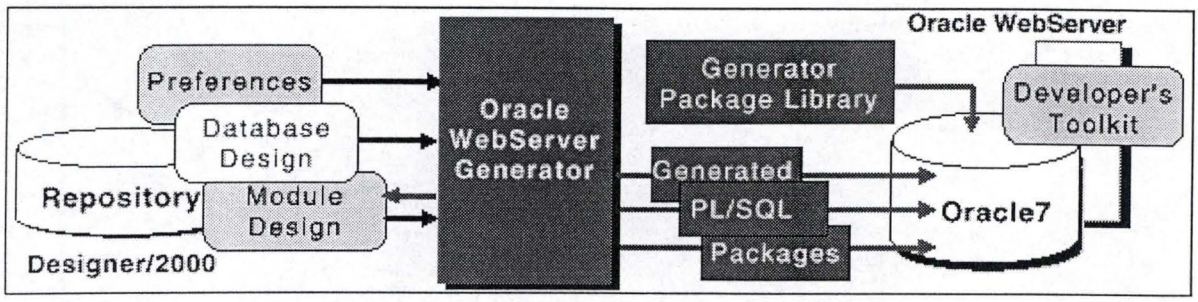


Figure 1-9 : Fonctionnement du WebServer Generator

Design des modules

Pour générer une application Oracle WebServer, l'utilisateur doit premièrement spécifier un ensemble de modules qui forment l'application et définir les liens entre ceux-ci. Le design de ces modules fournit la principale entrée du processus de génération.

Processus de génération

Quand le concepteur crée son application, il choisit quel module doit être généré et s'il faut ou non générer tous les modules appelés par le premier. L'Oracle WebServer Generator examine les spécifications de design de chaque module, détecte les liens entre ceux-ci et détermine les valeurs de la configuration de l'utilisateur.

La génération donne lieu à une série de fichiers et l'utilisateur a le choix d'installer l'application ainsi produite au sein de la base de données Oracle. Ensuite, l'utilisateur peut exécuter son application en utilisant un navigateur HTML.

Définition des modules et utilisation de données

Une application Oracle WebServer comprend une ou plusieurs pages HTML dynamiques. Chaque page générée contient une ou plusieurs sections, du texte HTML statique et des données formatées provenant d'une requête SQL.

Les composants des modules forment la base de chaque requête SQL et sont définis graphiquement en utilisant le module Data Diagrammer. Une définition de module peut contenir un ou plusieurs composants de module liés entre eux par une foreign key. Les composants de module sont à leur tour basés sur une table spécifique (table de base) mais peuvent aussi utiliser d'autres tables pour afficher de l'information (table lookup).

L'exemple ci-dessous comprend deux composants de module, l'un basé sur la table des départements d'une entreprise, l'autre sur la table des employés. Pour chaque composant de module, on doit déterminer les colonnes qui doivent être incorporées dans l'application et leurs caractéristiques d'affichage (l'intitulé, le formatage HTML, ...).

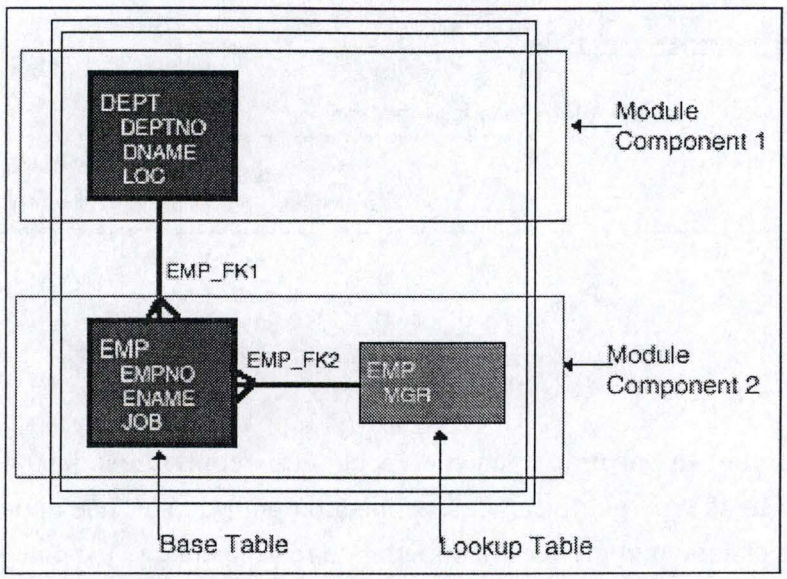


Figure 1-10 : Découpe en modules et composants de modules

Pages générées

Les pages générées par l'Oracle WebServer Generator peuvent être de quatre types : les « Startup Page », les « Query Form », les « Record List » ou les « View Form ».

Une « Startup Page » est créée pour chaque module de l'application générée. Un « Query Form » permet à l'utilisateur d'entrer des critères de recherche pour exécuter une requête. Une « Record List » affiche un ensemble de données provenant d'une requête dans une table. Enfin, une « View Form » affiche les détails complets pour un enregistrement donné. Le diagramme suivant montre la relation entre les différents types de page.

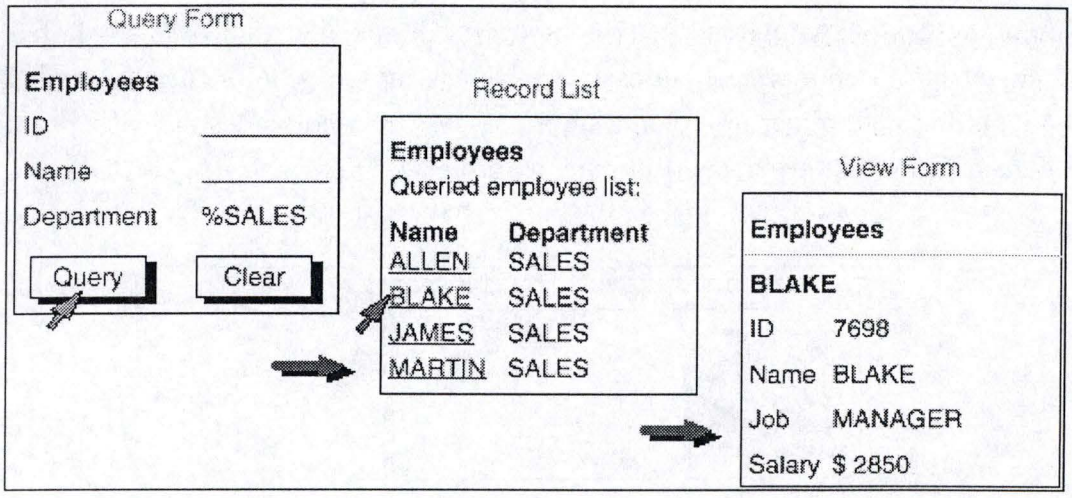


Figure 1-11 : Différents types de pages générées

Préférences utilisateur

L'apparence d'une application générée peut être configurée par l'utilisateur. La configuration peut se faire au niveau de l'application pour donner une apparence uniforme, ou au niveau de chaque module. Les paramètres se regroupent en cinq catégories : « DBA » qui comprend entre autres les paramètres liés à la base de données comme le nom de l'utilisateur, « General Layout » image de fond d'écran et séparateur de liste ; ainsi que « Query Form », « Record List Layout » et « View Form » comprenant l'alignement vertical ou non, formatage des données en table, en liste ou autre.

1.2.2.2 EKTOS

EKTOS comme base de données orientée objet. EKTOS [EKTOS97] est un logiciel de gestion de base de données orientée objet, créé et développé par la société belge EKTELIS. Typiquement, EKTOS permet de gérer des banques de données comme des catalogues de produits, d'images, d'œuvres d'art ou de mobilier de bureau. Il propose deux moyens d'accéder à l'information : d'une part, la navigation à travers toute l'information disponible, exactement comme on le ferait en feuilletant un catalogue imprimé et d'autre part, via un accès direct à un produit bien déterminé à l'aide de requêtes.

Une application EKTOS est définie par une structure hiérarchique de classes avec la facilité d'héritage. Chaque classe est constituée d'attributs (champs) et de liens vers d'autres classes ou vers les objets « AnyFile »[14] d'EKTOS.

[14] Objets génériques pouvant contenir n'importe quel type de données

Dans une application EKTOS, chaque lien peut être utilisé comme chemin de navigation. Puisque les liens sont à double sens, aucune direction ne doit être prédéfinie par le programmeur. Chaque instance (ou fiche) est un objet autonome et est construite dynamiquement avec l'information disponible.

Nous verrons par la suite qu'EKTOS possède une option qui le transforme en serveur Web (pour les données de la base) sans configuration ni programmation supplémentaire.

EKTOS comme outil de développement. La structure de la base de données est construite dynamiquement en utilisant une interface graphique. Les couleurs indiquent si l'attribut est créé (noir), hérité (gris), ou si l'attribut hérité a été modifié localement (vert). Les classes et les attributs peuvent être déplacés par « drag & drop » et toutes les données liées suivront automatiquement (si c'est logiquement possible). Cette construction est simple, intuitive et ne demande pas une grande connaissance des méthodes de conception de bases de données.

Par défaut, chaque instance est vue comme d'une fenêtre dynamique, les attributs vides ou non autorisés ne sont pas visibles. La présentation graphique de chaque classe peut être personnalisée.

Un langage de programmation événementiel permet en outre d'automatiser certaines fonctions. Les scripts peuvent être intégrés dans la base de données même ou être déposés dans une application pour une exécution immédiate.

Tous les fichiers localisés sur un support accessible peuvent enfin être représentés par un objet dans une base de données EKTOS. Un double click sur cet objet ouvre le fichier et l'application qui lui est associée. Chaque objet « AnyFile » contient les propriétés du fichier (la taille, les dates de création et de modification, les chemins relatifs et absolus, par exemple), le nom du volume où le trouver et une imagette lorsque le format le permet (par exemple pour les images et les vidéos).

EKTOS et sa structure client/serveur. Le serveur EKTOS contient l'application entière, c'est à dire la structure et ses données. Lorqu'il n'est pas utilisé, le client EKTOS ne contient toutefois rien de l'application. C'est seulement lorsqu'il se connecte au serveur qu'il reçoit l'application, les scripts et les données.

Un serveur EKTOS peut simultanément fournir des clients EKTOS et des navigateurs Web par Internet/Intranet et des réseaux propriétaires. Lorsque l'option Web est cochée dans la configuration du serveur, celui-ci génère des pages HTML à la demande. Il peut également exporter le contenu de sa base de données vers des pages HTML statiques.

Prenons l'exemple d'un catalogue de voitures. Une fiche représente un modèle de voiture. On peut y voir une photo de la voiture considérée, et une multitude d'informations comme les caractéristiques du moteur, les options, ses mesures, le prix, ... Si l'application le permet l'utilisateur peut par exemple cliquer sur le champ de l'option « toit ouvrant » pour obtenir le descriptif de cette option. Sur cette fiche de description, il peut y avoir un lien vers d'autres modèles de voitures ; l'utilisateur peut alors naviguer vers un autre modèle possédant cette option et ainsi de suite.

On peut imaginer l'équivalent Web de cette navigation : les liens existant implicitement dans EKTOS sont facilement transposables en HTML. Il existe cependant encore quelques limitations comme l'impossibilité de modifier la présentation des pages, présentation qui reste alors statique. De même, pour une application EKTOS normale, le concept d'image cliquable est simulé en déposant des objets sur une image de fond, mais il n'existe pas de transposition de cette notion vers le Web.

1.3 Méthodologies de développement

La représentation d'un problème constituera une partie conséquente de notre exposé. Un problème peut être décrit via un modèle de repésentation tel que le schéma ERA. Le concepteur, quant à lui, a sa propre représentation en terme d'implémentation dépendant du type de système utilisé, par exemple un fichier HTML décrivant un formulaire. Enfin, l'utilisateur a sa propre vue du problème et l'interface lui propose le moyen d'interagir.

Ces différentes représentations, respectivement appelées représentation interne, conceptuelle et externe, se retrouvent dans la méthodologie DIANE. Nous proposons une brève description de celle-ci et des différentes notions de représentation employées à la première sous-section.

Dans les deux sous-sections suivantes, nous décrivons deux méthodologies de développement d'applications hypermédias, RMM (Relationship Managment Methodology) et SHDT (Structured Hypermedia Design Technique). Ces deux méthodologies ont étés retenues car elles sont significatives par leur contenu. Chacune de ces méthodologies est abordée de trois points de vue : l'aspect modèles utilisés pour décrire l'environnement, l'aspect démarche définissant les règles à suivre et l'aspect des outils susceptibles d'être utilisés comme support. Ces méthodologies sont analysées et utilisées sur un cas concret au chapitre suivant.

D'autres méthodologies comme EORM [LANGE94], OOHDM [SCHWABE95] et SOHDM [SOHDM] sont également proposées, mais leur étude et le peu d'informations disponibles à leur propos ne nous ont pas permis de porter un jugement efficace quant à leur utilisation. Cependant, nous renvoyons le lecteur en Annexe D pour une brêve description de ces autres méthodolgies permettant le développement d'applications hypermédias.

1.3.1 Les différentes représentations de la méthodologie DIANE

DIANE est une méthodologie de développement d'applications interactives s'appuyant sur une autre méthodologie appelée MERISE. Elle concerne les phases de spécification et de conception de l'IHM (Interface Homme-Machine) d'une application en reprenant de MERISE les concepts de tâche, d'opération et en les raffinant dans une perspective d'IHM. MERISE est la méthodologie de génie logiciel de référence en France pour développer un système d'information. Elle comprend une modélisation complète de la base de données sur laquelle travaillent les fonctions sémantiques du noyau fonctionnel. Cette méthodologie prévoit aussi une description de l'organisation abritant le système d'information et ses liens avec les postes de travail des utilisateurs.

DIANE intègre dans MERISE l'ergonomie logicielle dans une perspective d'IHM. Pour réaliser cette intégration, elle propose un modèle de l'application interactive exprimé suivant trois points de vue :

1.3.1.1 La vue du programmeur

Par une **représentation interne**, elle décrit la logique fonctionnelle des traitements informatiques du SI et de leurs effets. Cette représentation est une vue purement technique du SI, c'est pourquoi aucune spécification relative à l'utilisateur n'est présente.

1.3.1.2 La vue du concepteur de l'interface

Par une **représentation conceptuelle**, elle décrit la logique d'utilisation de l'interface. Cette représentation se veut conforme à la représentation mentale que possède l'utilisateur de son SI : c'est une vue purement opérationnelle du SI puisqu'aucune considération technique relative au SI n'intervient. La représentation externe vient compléter la logique fonctionnelle de la représentation interne. Pour y aboutir, DIANE interprète un SI comme un ensemble de postes de travail. Chaque poste de travail offre à l'utilisateur la faculté d'accomplir un ensemble donné de tâches interactives pour remplir le rôle qu'il doit assumer au sein de l'organisation.

1.3.1.3 La vue de l'utilisateur

Par une **représentation externe**, elle concrétise les éléments modélisés de la vue conceptuelle en s'appuyant sur des connaissances substantielles de l'ergonomie logicielle, par exemple des critères ergonomiques de conception, des règles ergonomiques. Elle doit aussi tenir compte des contraintes techniques provenant de l'environnement physique. Il existe pour DIANE des règles de conception permettant de dériver systématiquement une représentation externe à partir d'une représentation conceptuelle. DIANE+ est l'outil de support à cette génération automatique.

1.3.2 Relationship Management Methodology (RMM)

RMM [ISAKOWITZ94] [ISAKOWITZ95a] (ou méthodologie de gestion de relations) est composée d'un modèle de données de gestion relationnelle (RMDM) et de sept phases de modélisation que nous explicitons ci-après. Nous décrivons ensuite RMCase [ISAKOWITZ95b], outil de support à cette méthodologie.

1.3.2.1 Relationship Management Data Model (RMDM)

Un modèle de données est un ensemble d'objets logiques représentant une partie du monde réel. Une des caractéristiques principales du modèle RMDM est qu'il offre un langage permettant de décrire les objets informationnels et le mécanisme de navigation caractérisant les applications hypermédias.

Ce modèle se base sur un schéma entité-relation. Le langage est caractérisé par des primitives de différents types qui possèdent chacune une représentation graphique. Le premier type regroupe les primitives concernant les Entités-Relations (ou ER) : celles-ci expriment la manière de structurer l'information dans le domaine d'application. Le deuxième type comprend les « slices » (tranches) et modélise la manière dont les informations sont présentées à l'utilisateur. Le dernier type correspond aux primitives d'accès, modélisant la navigation entre les informations.

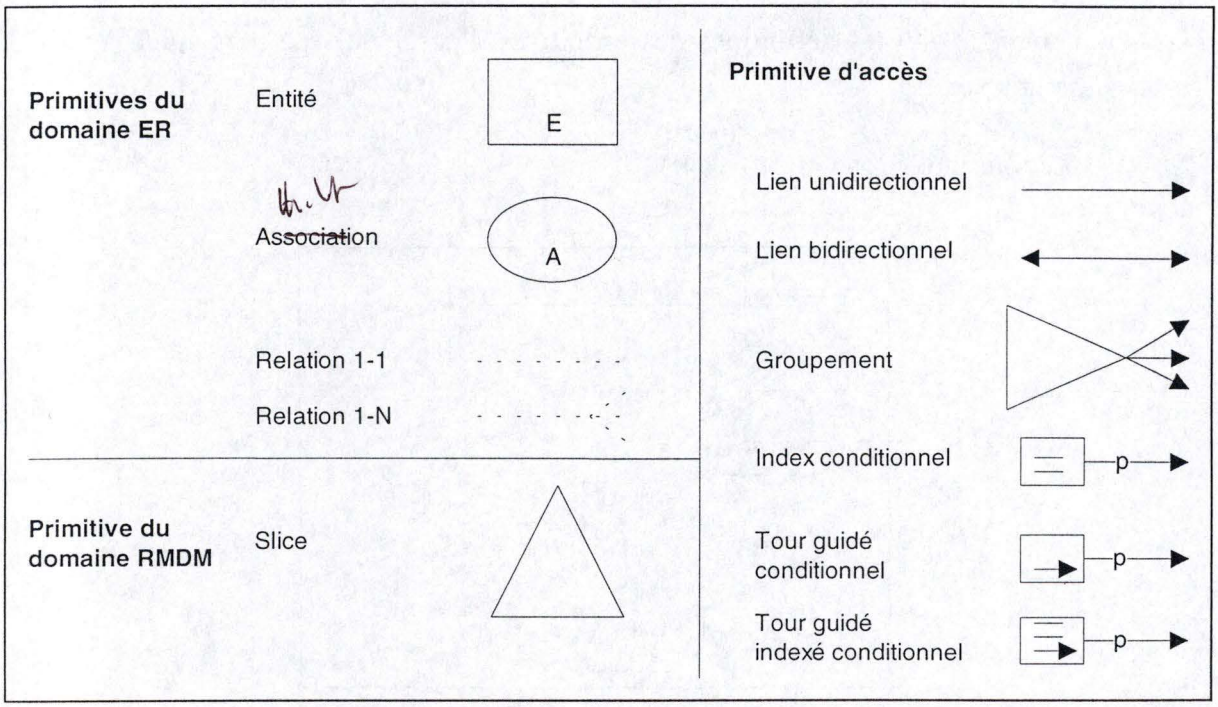


Figure 1-12 : RMDM Relationship Managment Data Model

Les primitives ER reprennent les types d'entités ainsi que leurs attributs (représentant les objets abstraits ou réels) et les associations (décrivant les associations entre les différents types d'entités)[15]. La notion de tranches a été créée car il n'est pas toujours confortable de présenter en une fois tous les attributs d'une même entité à l'utilisateur. Ainsi, lorsqu'une entité « personne », possédant les attributs « nom », « âge », « photo » et « biographie », sera modélisée, il sera plus convivial de découper les informations en une tranche générale concernant le nom, l'âge et la photo et en une autre tranche concernant la biographie.

Enfin, les primitives d'accès peuvent représenter des liens uni- ou bidirectionnels, afin de modéliser les accès entre les tranches d'une entité[16], des index, des tours guidés implémentant un chemin linéaire à travers une collection d'éléments et permettant à l'utilisateur d'avancer ou de reculer dans son exploration, ainsi qu'un regroupement fournissant un mécanisme de type menu, pouvant par exemple proposer à l'utilisateur le type d'exploration souhaitée.

[15] Comme dans la modélisation des bases de données, les associations seront de types un-un ou un-plusieurs.
[16] C'est ainsi que ces liens ne peuvent représenter qu'une navigation au sein d'une seule entité.

Graphiquement, voici les définitions des primitives d'accès de type tour guidé, index et tour guidé indexé :

- Le tour guidé (ou Guided Tour)

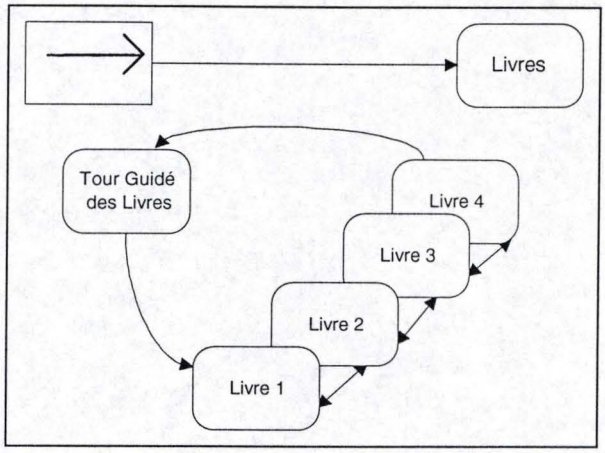


Figure 1-13 : Le tour guidé

Un tour guidé consiste donc à parcourir une liste d'entités en partant d'une première, séquentiellement jusqu'à la dernière, avec la possibilité de marche arrière.

- L'index

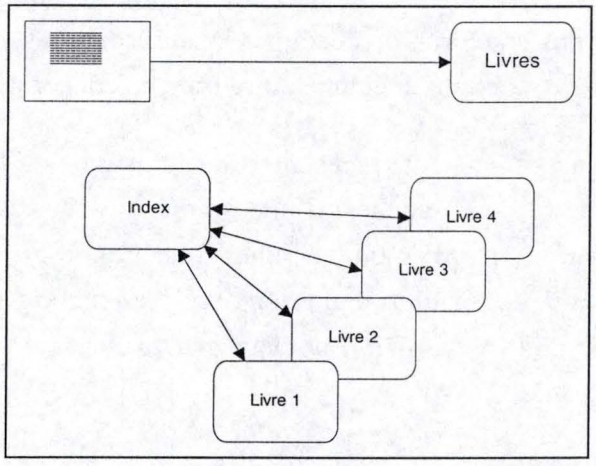


Figure 1-14 : L'index

Un index propose un menu permettant de choisir une entité à visiter ; l'utilisateur peut ensuite revenir à l'index.

- Le tour guidé indexé

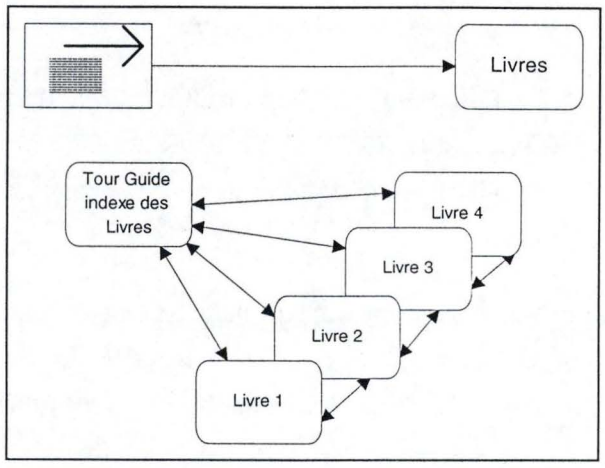


Figure 1-15 : Le tour guidé indexé

Le tour guidé indexé combine le tour guidé et l'index. L'utilisateur peut donc choisir une entité de départ pour un tour guidé et revenir à l'index à n'importe quel moment de la navigation.

Enfin, les prédicats de conditions qualifiant un index ou un tour guidé déterminent quelles instances sont accessibles. Par exemple, le prédicat Livre(Auteur='Dupont') imposé à un tour guidé précise que seuls les livres dont l'attribut Auteur est égal à Dupont seront accessibles.

1.3.2.2 Etapes de la démarche de la méthodologie RMM

La méthodologie RMM se concentre sur les phases de design, de développement et de construction. En particulier, la phase de design utilise le modèle RMDM et se compose de trois étapes : la procédure de design ER, la découpe en tranches et le design navigationnel.

La procédure de design ER. L'objectif est de représenter le domaine d'information de l'application à travers un schéma entité-relation. Les entités et les relations ainsi construites constituent la base même d'une application hypermédia et beaucoup d'entre elles seront représentées dans l'application finale en tant que liens ou nœuds[17].

On obtient comme résultat le schéma ER représentant le système d'information propre à l'application.

[17] Page centrale servant de point de départ pour une navigation éventuelle.

La procédure de découpe en tranches (ou design d'entités). Cette procédure a pour but de déterminer, dans les entités choisies, la manière dont les informations seront représentées et accessibles pour l'utilisateur.

Un diagramme de répartition (slice diagram) est produit et chaque tranche (slice) regroupe un ou plusieurs attributs d'une entité. On possède alors un schéma ER enrichi, noté ER+, obtenu à partir du premier schéma où chaque entité est complétée par son diagramme de répartition.

Remarquons que chaque entité possède une tranche spécifiée par un nom d'en-tête qui servira par la suite comme ancre afin d'accéder à l'entité. Les relations entre tranches peuvent aussi être modélisées à l'aide de liens uni- ou bidirectionnels qui portent alors un nom.

La procédure de design navigationnel. L'objectif est ici de modéliser les chemins qui rendront la navigation possible. Chaque relation présente dans le schéma ER+ est analysée et une relation est remplacée par une ou plusieurs structures d'accès RMDM. Il est ensuite nécessaire de regrouper certains éléments afin d'obtenir une structure d'accès de plus haut niveau. Ce groupement fournit alors un accès de type menu. Dans cette étape, le designer doit non seulement identifier le contenu des informations, les relations qui seront accessibles, mais aussi distinguer les groupements présents et les structures d'accès utilisées dans chaque cas.

Cette troisième procédure élabore un diagramme RMDM complet, dit schéma navigationnel, résultant de la transformation du schéma ER+. Les structures d'accès devant être implémentées sont ainsi décrites.

Les autres étapes. Les trois étapes principales de la méthodologie RMM ont été décrites ci-dessus. Au total, elle en comporte sept. Nous décrivons brièvement les autres phases dans ce paragraphe. L'étape quatre, appelée protocole de conversion, utilise un ensemble de règles de transformation pour chaque élément du modèle RMDM, afin d'obtenir un équivalent pour la plate-forme de destination (par exemple une page HTML). L'étape cinq concerne l'interface utilisateur et s'occupe entre autres d'améliorer la présentation de l'IHM. L'étape suivante ou « runtime behavior design » gère des éléments d'exécution comme l'historique des liens, du retour en arrière etc. Finalement, l'étape sept, assez traditionnelle, est centrée sur les tests de l'application réalisée.

Notons que pour cette méthodologie, le développement d'une application hypermédia est basé d'avantage sur une méthode « bottom-up » : l'attention est portée en premier lieu sur chaque entité, pour ensuite remonter à une structure d'accès plus générale.

1.3.2.3 Outil de support

L'outil supportant la méthodologie est RMCase, il permet le design d'applications hypermédias pour le Web. Il se base pour cela sur la méthodologie RMM et son modèle de données RMDM. Il supporte les trois phases essentielles de design (production d'un schéma ER, découpe en tranches et production du schéma navigationnel) tout en proposant à l'utilisateur une interface graphique permettant l'édition des schémas et reprenant les concepts de RMDM. Ci-dessous, une capture d'écran du programme pour chacune de ces trois phases :

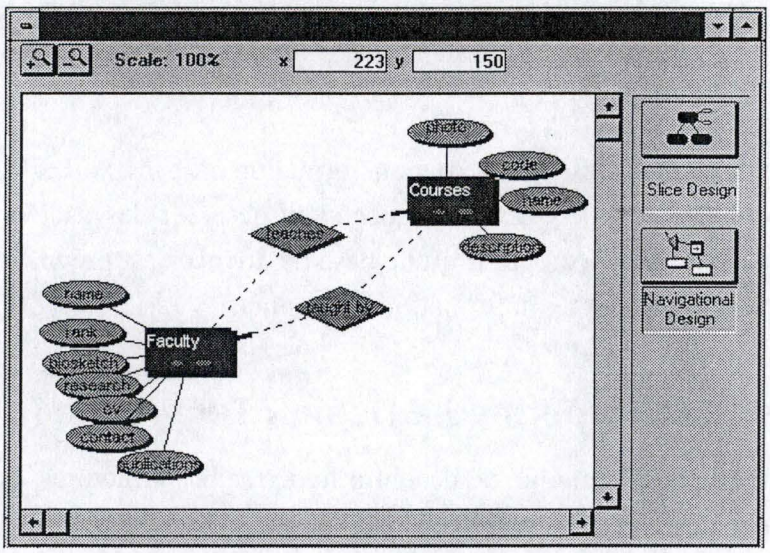


Figure 1-16 : Design ER avec RMCase

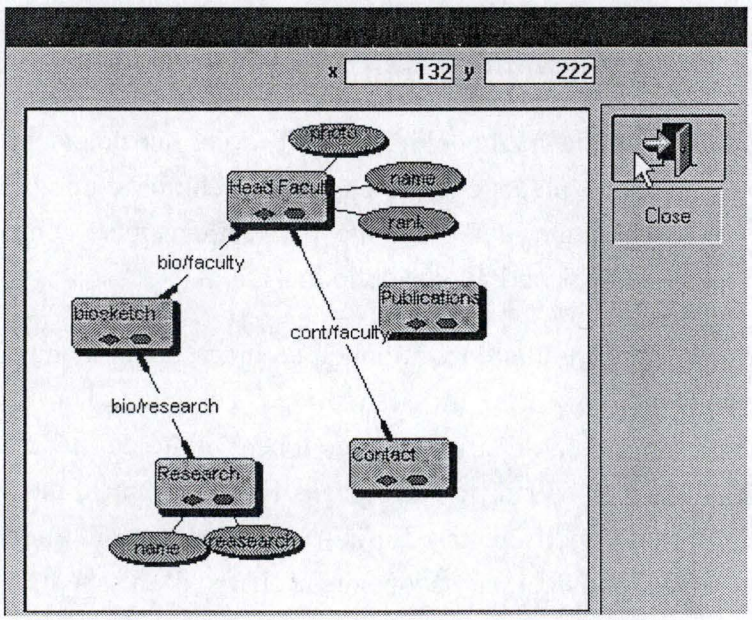


Figure 1-17 : Découpe en tranches avec RMCase

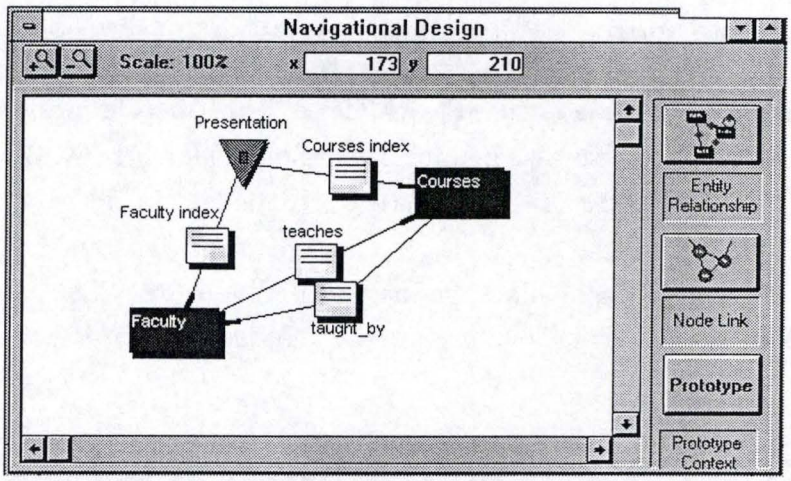


Figure 1-18 : Design navigationnel avec RMCase

L'outil supporte également les phases de transformation en syntaxe HTML mais de façon passive pour l'utilisateur. Une extension s'occupant de créer des canevas (ou templates) pour la visualisation de valeurs à partir d'une base de données est prévue, mais il ne nous a pas été possible d'obtenir assez d'informations à ce sujet.

1.3.3 Structured Hypermedia Design Technique (SHDT)

SHDT [BICHLER96] (ou Technique de design d'hypermédias structurés) est une technique semi-formelle de design de sites Web, composée de deux phases principales : la structuration de l'information et le design navigationnel. Elle est implémentée dans un outil appelé Web Designer. Nous allons voir que celui-ci, utilisant le petit nombre d'objets de la méthodologie, génère automatiquement les pages HTML et les scripts CGI adaptés.

1.3.3.1 Modèle de représentation et structuration de l'information

Le méta-modèle utilisé par la méthodologie SHDT a été spécialement conçu pour être simple, un des buts des concepteurs était de parvenir a réduire le nombre d'objets utilisés par le modèle. Pour la technique SHDT, un site Web est considéré comme étant composé d'au moins une page et de liens ou Layouts optionnels.

Une page peut être soit un formulaire traditionnel, un index ou un menu. Ces primitives ou objets statiques peuvent aussi être utilisés comme Template. Un Template ou objet dynamique peut alors être assimilé au concept de type d'entité du modèle ERA. Un même Template est utilisé pour générer différentes pages HTML ayant la même structure, mais comportant des informations différentes. Enfin, un diagramme détermine un raffinement hiérarchique ; son but est d'alléger par découpage la charge d'un schéma.

L'objectif de la phase de structuration de l'information est de repérer les primitives à utiliser pour représenter les éléments de structure du future site Web.

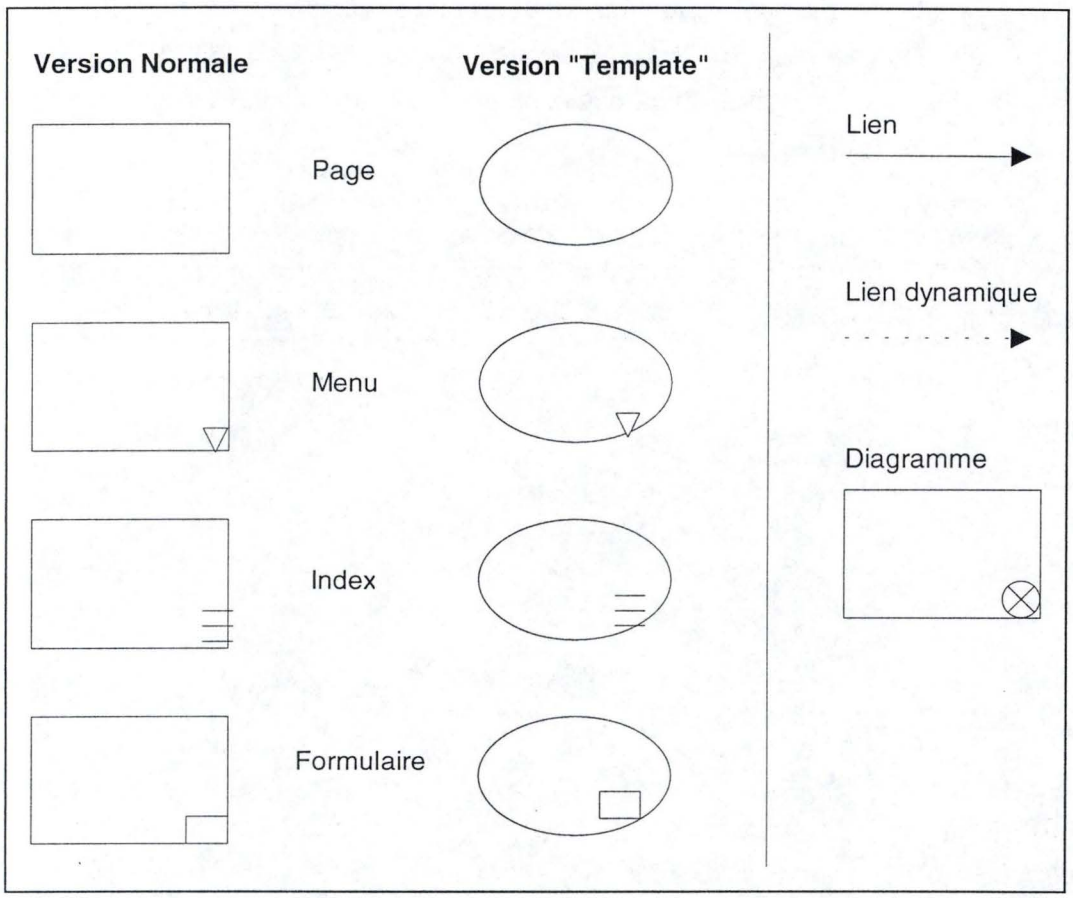


Figure 1-19 : Modèle de donnée de SHDT

1.3.3.2 Design navigationnel

Les liens statiques et les liens dynamiques sont utilisés dans le modèle navigationnel. Le premier est le lien standard pointant, par exemple, vers une page HTML bien précise. Le second est appelé dynamique car il « pointe » vers une page qui sera générée « on the fly », c.-à-d. à l'exécution, par un script CGI. Pour donner un parallèle avec le paragraphe précédent, les liens statiques pointent vers des objets statiques, tandis que les liens dynamiques pointent vers des Templates.

L'objectif de cette seconde phase est donc de relier par les primitives adéquates les éléments structurels repérés précédemment.

1.3.3.3 Outil de support

Les phases de structuration de l'information et de design navigationnel sont implémentées par Web Designer en une application graphique simple à utiliser. La Figure 1-20 donne une idée de l'interface proposée. Elle génère alors les différentes pages HTML et scripts CGI nécessaires à la recherche d'informations dans une base de données et/ou à la génération automatique de pages HTML.

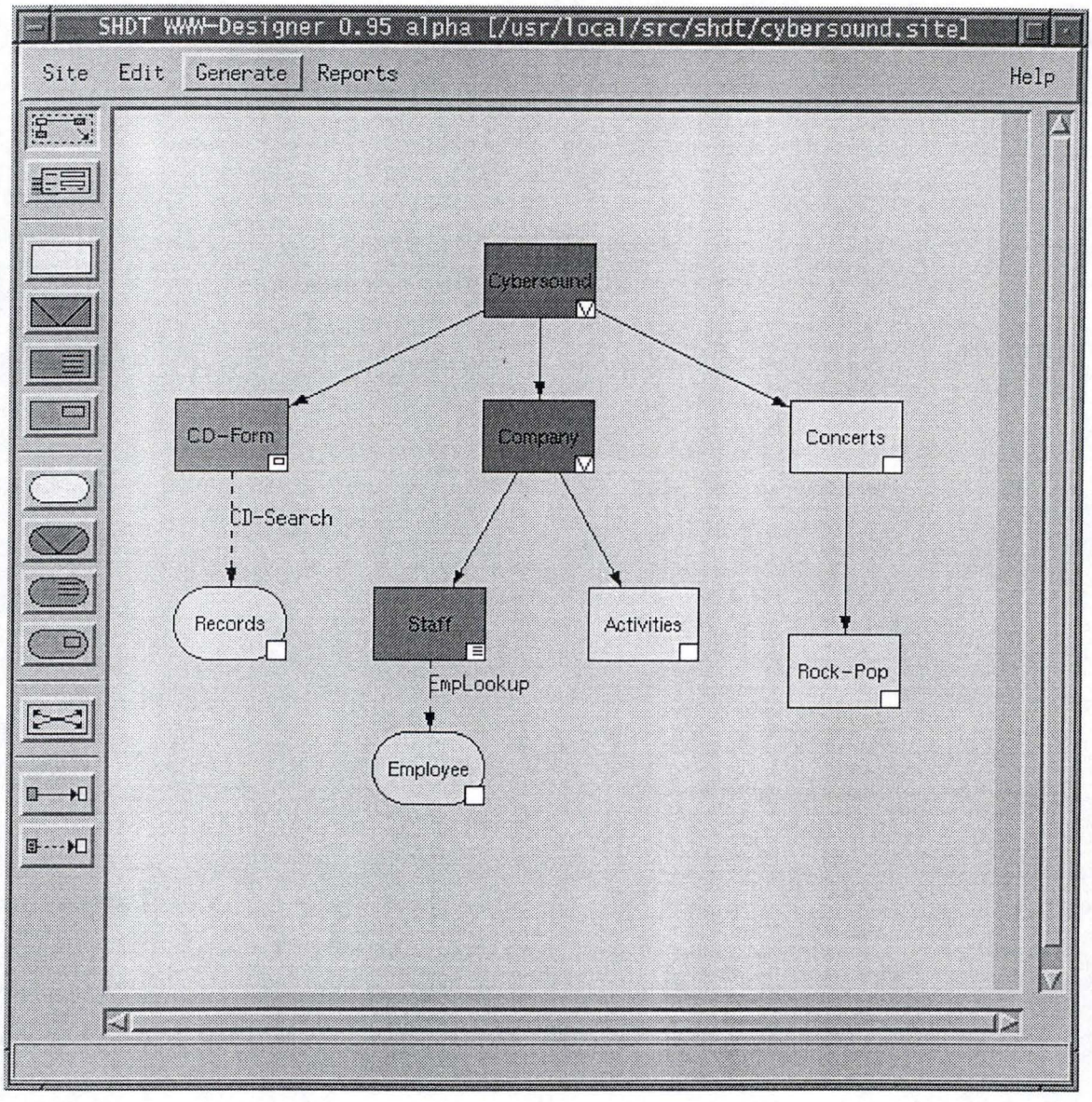


Figure 1-20 : Procédure de design avec SHDT

Il est à noter que la production du programme est un squelette de pages HTML éditables par n'importe quel outil adapté (éditeur HTML pour le premier, éditeur de texte en général). Le concept de Layout étant proche de celui de Style-Sheets[18], Web Designer permettra alors la création de pages uniformes pour tout le site.

[18] Proposition du consortium W3 ayant pour but de définir une présentation commune de pages HTML en se basant sur des canevas réutilisables.

Chapitre 2. Développement de formulaires simples

Dans ce deuxième chapitre, après un descriptif de notre première classe de problèmes c'est-à-dire la création d'applications composées de formulaires simples, nous étudions le contexte de la tâche à l'aide de la méthodologie Trident afin d'en déduire un style d'interaction. Ensuite, nous étudions et observons sur un cas concret les différents apports des méthodologies et/ou outils décrits dans le premier chapitre. Une comparaison de ceux-ci nous amène à en donner les points forts et faibles, et l'observation finale nous conduit alors à élargir notre vue du problème.

...ons simples utilisant des bases de données, s'appuyant à leur tour sur ...nceptuel. Ce dernier est composé d'attributs de type simple (entier, chaîne de ...ères, booléen, etc.).

Précisons que les applications en question sont destinées dans notre cas au monde particulier du World Wide Web. Dans ce cadre peuvent être intégrés les programmes du type : recherche simple dans des catalogues et/ou index de documents, de personnes ou encore consultation d'ouvrages appartenant à une bibliothèque.

Afin de circonscrire ce contexte, nous fixons les paramètres relatifs à celui-ci dans les sous-sections suivantes. Nous en dérivons un style d'interaction à l'aide des tables de dérivation de la méthodologie Trident [VANDERDONCKT93].

2.1.1 Analyse de la tâche

Une *tâche* est définie comme une activité dont l'accomplissement par un opérateur produit un changement d'état significatif d'un domaine d'activité donné, dans un contexte donné.

2.1.1.1 Pré-requis

Définition : les pré-requis d'une tâche expriment la quantité de connaissances du système existant/futur que l'utilisateur doit posséder en vue de l'utilisation efficace du système. Les pré-requis peuvent notamment traduire la formation nécessaire et peuvent donc être minimaux, modérés ou maximaux.

La tâche étant simple, les pré-requis sont *minimaux*.

2.1.1.2 Productivité

Définition : la productivité d'une tâche représente le nombre moyen d'exécutions de la tâche par unité de temps. Elle peut aussi recouvrir la fréquence d'utilisation. La productivité peut être faible, moyenne ou élevée.

Faible. Utilisation « à l'occasion ». Dans notre exemple, l'utilisateur fera une recherche d'un ou de quelques ouvrages, sans plus. Une fois son but atteint, il ne réutilisera peut-être le système qu'après un laps de temps assez long. Il ne s'agit donc en aucun cas de prévoir une utilisation « intensive » voir même journalière ou à intervalles réguliers de la tâche à effectuer.

2.1.1.3 Environnement objectif

Définition : L'environnement de la tâche traduit la présence (environnement dit existant) ou l'absence (environnement dit inexistant) d'objets spécifiques manipulés dans le cadre de l'accomplissement de la tâche, indépendamment de tout système.

Existant. Il existe par exemple des index dans les bibliothèques, des catalogues papiers de fournitures, etc.

2.1.1.4 Reproductibilité de l'environnement

Définition : La reproductibilité de l'environnement d'une tâche est réalisable ou non pour autant que cet environnement existe. La reproductibilité de l'environnement est dite praticable si elle peut être transposée dans le cadre du système et non-praticable dans le cas contraire.

Praticable. Le système est la transposition informatique d'un catalogue ou index existant dans le monde réel avec les mêmes possibilités de recherche et de consultation (via des index ou des tables des matières par exemple).

2.1.1.5 Structuration de la tâche

Définition : La structuration de la tâche explique le degré de liberté ou de contrainte que l'utilisateur a dans son accomplissement. La structuration de la tâche peut être faible, modérée ou élevée.

Elevée. En effet, la marge de liberté laissée à l'utilisateur est très faible, comme nous allons le voir par la suite, le modèle d'interaction proposé suit un schéma qui ne peut être élargi par l'utilisateur.

2.1.1.6 Importance de la tâche

Définition : L'importance de la tâche informe quant au caractère fondamental, crucial, vital d'une tâche. L'importance de la tâche peut être faible, modérée ou élevée.

Faible. Dans la plupart des cas, la tâche n'est pas vitale pour l'utilisateur : elle consiste en une consultation de fiches, qui dans le cas de petites bases de données pourraient même être remplacées par un fichier papier.

2.1.1.7 Complexité de la tâche

Définition : La complexité d'une tâche manifeste le degré de complexité cognitif, manipulatoire, intellectuel quant aux sous-tâches et actions mises en jeu en vue d'accomplir la tâche. Nous pouvons trouver des tâches dont la complexité est faible, modérée ou élevée.

La complexité de la tâche décrite est *modérée.*

2.1.1.8 Première dérivation

Suite à l'analyse de ces paramètres relatifs à la tâche analysée, on peut dériver un style d'interaction à l'aide des tableaux de prise de décision définis par la méthodologie Trident. Cette dérivation se veut uniquement un choix de départ qui ne deviendra définitif qu'au niveau de chaque unité de présentation. De cette première étape nous pouvons dériver les styles d'interaction de type questions/réponses et remplissage de formulaires.

2.1.2 Type d'utilisateur

L'utilisateur est « monsieur tout le monde », pour autant qu'il ait accès à un ordinateur de type PC (personnel ou via une unité de service) et désirant, par exemple, consulter les ouvrages d'une bibliothèque. La tâche étant particulièrement simple à effectuer, l'accès en sera facilité.

2.1.2.1 Expérience de la tâche

Définition : Explique les connaissances syntaxiques et sémantiques relatives à la tâche. Les connaissances syntaxiques touchent à la position de la tâche, son insertion dans le cadre complet du système ainsi que la terminologie utilisée. Les connaissances sémantiques ont trait aux objets, règles et procédures de gestion impliquées dans la tâche. L'expérience de la tâche peut être élémentaire, moyenne ou riche.

Généralement *moyenne*. L'utilisateur doit connaître certaines données relatives au domaine d'application. Par exemple, la personne doit entre autres connaître l'existence du numéro ISBN d'un livre, son but, son utilité, sa syntaxe.

2.1.2.2 Expérience du système

Définition : L'expérience du système donne une idée du niveau d'expérience de l'utilisateur en matière d'utilisation de systèmes d'information, qu'ils soient dédiés à la tâche concernée ou non. Elle peut être élémentaire, moyenne ou riche.

Egalement *moyenne* et concerne les rudiments de l'utilisation d'index et du système des pages Web.

2.1.2.3 Motivation

Définition : La motivation est censée traduire l'attitude psychologique vis-à-vis de la tâche à réaliser. Si l'utilisateur est enclin (respectivement peu ou moyennement enclin) à accomplir la tâche, sa motivation sera élevée (respectivement faible ou modérée).

Modérée. Cela dépend en fait de l'utilité de la recherche effectuée par l'utilisateur ainsi que des performances du système.

2.1.2.4 Expérience d'un moyen d'interaction complexe

Définition : Cette expérience rend compte de l'aptitude, de l'expérience qu'a l'utilisateur de manipuler un ou plusieurs moyens d'interaction complexes tels que le clavier étendu, un tableau de commande complet, etc. Elle peut être élémentaire, moyenne ou riche selon les cas.

Elémentaire. Cette expérience n'est pas requise pour ce type d'application.

2.1.2.5 Seconde dérivation

Les tables d'analyse des paramètres relatifs à l'utilisateur potentiel confortent notre position quant au choix du style d'interaction de type remplissage de formulaires et répond donc bien à notre attente concernant la tâche proposée.

2.1.3 Description de l'environnement

L'environnement dans lequel nous travaillons est le monde du World Wide Web composé de pages HTML et dont les applications ont les caractéristiques suivantes :

2.1.3.1 Type de traitement

Définition : Précise si l'utilisateur au poste de travail considéré ne peut effectuer que la tâche considérée ou plusieurs. Il peut être, selon les cas, mono- ou multi-traitement.

Multi-traitement. L'application s'exécutant sur une machine de type PC (compatible IBM, Macintosh ou autre), celle-ci n'est pas uniquement dédiée à la seule application concernée et l'utilisateur peut réaliser d'autres tâches en parallèle.

2.1.3.2 Capacité de traitement

Définition : La capacité de traitement considère le niveau d'interruptibilité, de parallélisme, de concurrence, d'interpénétrabilité des tâches qui peuvent être effectuées au poste concerné. La capacité de traitement peut être faible, moyenne ou élevée.

Moyenne. En effet, l'utilisation de l'application peut s'insérer dans un travail plus large. Dans notre cas, il peut par exemple être complémentaire à la création d'un document texte pour une bibliographie.

2.1.3.3 Troisième dérivation

De même, les tables de dérivation nous confirment l'efficacité du choix du style d'interaction reposant sur le remplissage de formulaires.

2.1.4 Attributs de dialogue

2.1.4.1 Contrôle du dialogue

Définition : Le contrôle du dialogue est interne lorsqu'il réside dans l'application, il est externe lorsqu'il est maintenu par l'interface et il est mixte lorsqu'il est tour à tour interne et externe.

Interne. En effet, pour l'utilisateur, l'introduction de valeurs qui serviront à la recherche du produit peut se faire de différentes façons, c'est donc lui qui possède le contrôle.

2.1.4.2 Mode de dialogue

Définition : Le mode de dialogue est séquentiel si le dialogue est basé sur des actions qui s'enchaînent directement. Il est asynchrone si l'ordre d'exécution de ces actions est purement non linéaire, non prédéterminé, non séquentiel. Il est mixte s'il est tour à tour séquentiel ou asynchrone.

Asynchrone. L'utilisateur n'est pas contraint de suivre un ordonnancement prédéfini.

2.1.4.3 Mode de déclenchement

Définition : Le type de déclenchement est soit automatique si le déclenchement de la fonction est à l'initiative du dialogue (par exemple pour les fonctions ne comportant que des informations internes en entrée), soit manuel si le déclenchement est à l'initiative de l'utilisateur. Dans ce cas, il peut être implicite si le déclenchement résulte d'une action non prévue à cet effet ou explicite si le déclenchement résulte d'une action de l'utilisateur prévue à cet effet. Dans ce dernier cas, il peut être soit affiché si le déclenchement se traduit par la présence d'objets interactifs concrets, soit non affiché si aucun objet dans la présentation ne laisse supposer le déclenchement.

Manuel explicite affiché. Bien souvent, l'utilisateur devra confirmer ses choix, valider les données introduites via un objet dédié à cet effet.

2.1.4.4 Métaphore

Définition : Ce paramètre concerne la transformation des objets sémantiques du monde réel en objets systèmes. Elle peut être de deux types : basée sur la conversation, si elle est basée sur une description linguistique des actions à accomplir ; basée sur le mini-monde, si elle mimétise électroniquement les objets du monde réel. Dans le premier cas, c'est l'utilisateur qui donne des commandes, des ordres (il parle à la troisième personne). Dans le second, l'utilisateur devient un acteur (il parle à la première personne).

Conversation. En effet, l'approche de cette première classe de problèmes se base sur des notions textuelles et linguistiques des actions à accomplir (par exemple, remplir le champ correspondant au titre du livre ou encore appuyer sur le bouton d'inscription).

2.1.4.5 Quatrième dérivation

Enfin, cette quatrième dérivation conforte également l'utilisation du style d'interaction de type remplissage de formulaires.

2.1.5 Conclusion

L'objectif est la conception d'une application composée de formulaires WEB simples qui répond aux tâches spécifiées ci-dessus pour les utilisateurs correspondants aux descriptions également reprises ci-dessus.

De cette étude nous pouvons effectuer la dérivation finale du style d'interaction propice au type d'applications dont les tâches ont été décrites ci-dessus. Le style est donc celui du remplissage de formulaires simples (composés uniquement d'éléments textes) décrits dans le langage HTML. On aura donc un environnement du type représenté par la Figure 2-1 (à titre indicatif) :

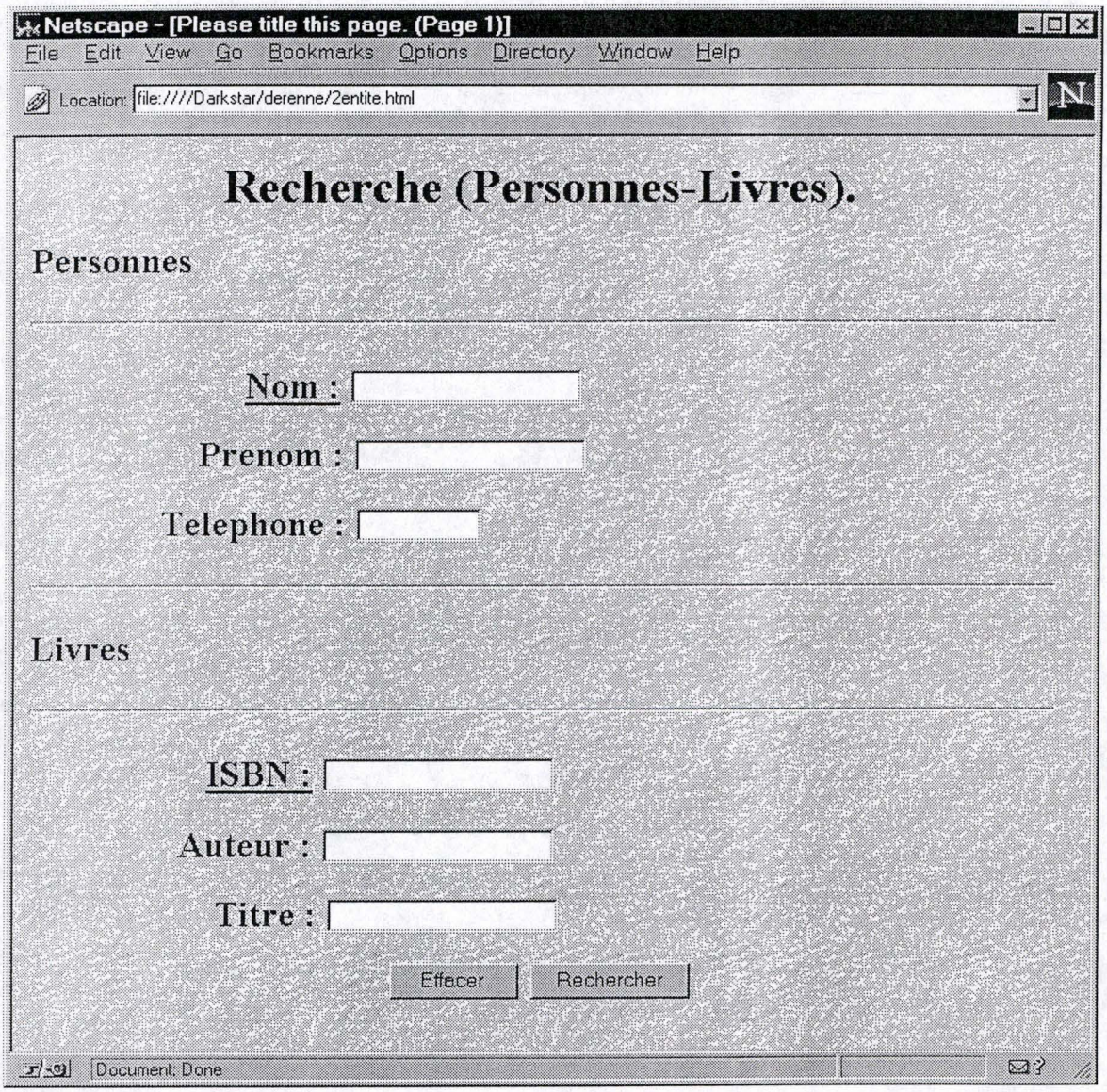


Figure 2-1 : Exemple de formulaire

La conception d'applications WEB, comme la conception de logiciels, est composée de trois éléments :

- Les *modèles*: tels que les modèles conceptuels, les diagrammes de flux, les graphes d'enchaînement, les modèles de la dynamique, de la statique, etc. Ceux-ci servent à décrire de façon abstraite la structure et la nature des données qui sont utilisées par l'application. Nous parlons alors de la vue interne du problème.

- Les *outils*: tels que les outils de modélisation ou de design d'interfaces. La production de ces outils est par exemple dans notre cas un fichier HTML. Ce fichier, qui ne sera vu que par le concepteur, constitue la vue conceptuelle du problème. Tandis que l'interprétation de ces données, sa visualisation finale via l'interface qu'il peut créer donne la vue externe, la seule visible par l'utilisateur final.

- Les *méthodes*: telles que les marches à suivre se basant sur certains modèles et/ou outils prédéfinis.

Par une application sur un cas concret des méthodologies décrites au premier chapitre, repérons au travers de ces trois notions les éléments susceptibles de nous aider dans la résolution de la première classe de problèmes.

2.2 Application des méthodologies sur une étude de cas : Emprunt d'un livre dans une bibliothèque

Nous examinons dans cette section et par une étude de cas (présentée par la première sous-section), les apports des méthodologies et outils dans la phase de conception de formulaires simples. Dans une seconde sous-section, nous lui appliquerons RMM et dans la dernière c'est SHDT qui est appliqué.

2.2.1 Présentation du cas

Nous avons choisi un exemple simplifié de la traditionnelle « bibliothèque ». Elle est décrite dans le schéma ERA suivant qui nous servira de base.

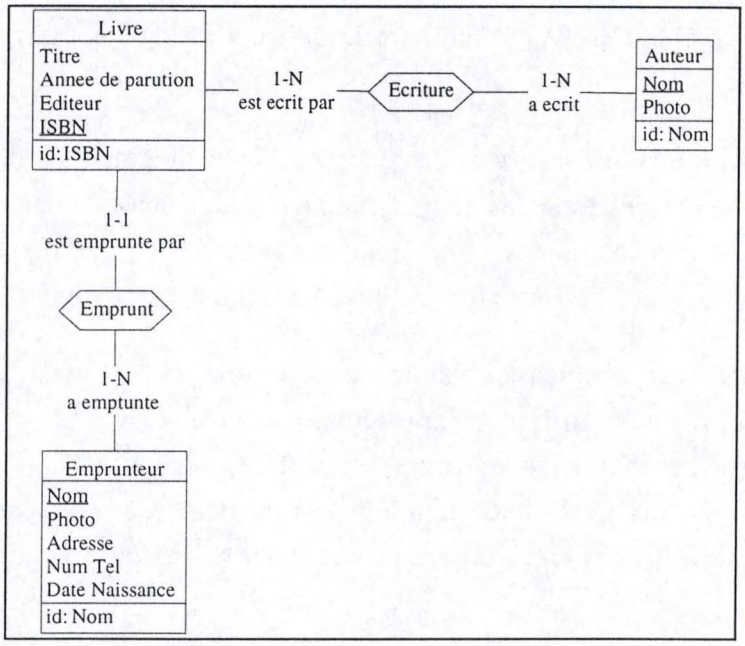


Figure 2-2 : Schéma ERA du problème de la « bibliothèque »

Un Livre est identifié par son numéro ISBN et est composé d'un titre, d'une année de parution et de son éditeur. Un Livre est écrit par un ou plusieurs Auteurs. Un Auteur est à son tour identifié par son nom et est caractérisé par sa photo ; un Auteur écrit un ou plusieurs Livres. Un Livre ne peut être emprunté que par un et un seul Emprunteur. Celui-ci est identifié par son nom et a comme caractéristiques sa photo, son adresse, son numéro de téléphone et sa date de naissance.

Il est à noter que le schéma et les entités en général se veulent simples et ne correspondent peut-être pas à un système réel. Les fonctions offertes (ou du moins que l'on voudrait offrir) à l'utilisateur sont de quatre types : la recherche, la consultation, l'ajout et la modification, et ce pour les différentes entités.

L'outil supportant RMM (cfr. 1.3.2) permet l'édition du schéma relationnel que l'on peut déduire du schéma ERA ci-dessous. De même, l'outil DBMain[19], développé à l'Institut d'Informatique de Namur, peut aider à la conception d'un tel schéma et même à sa traduction en un schéma relationnel.

Appliquons maintenant différentes méthodologies afin de repérer jusqu'où elles permettent de résoudre notre problème. Cet exemple a été appliqué à RMM et SHDT. Vu le peu d'informations disponibles pour les autres méthodologies, leur application ne s'avérait pas des plus aisées pour la création d'applications hypermédias.

[19] http://www.info.fundp.ac.be/~dbm

2.2.2 Application de la méthodologie RMM

2.2.2.1 Découpe en tranches

Les étapes de cette phase sont les suivantes :

1. Découpe en tranches : décrit le découpage des informations pour la présentation de celle-ci à l'utilisateur. Remarquons que, notre exemple étant simplifié, la découpe a été un peu influencée.

2. Choix de l'en-tête : une tranche est choisie comme en-tête. Dans notre cas, il s'agira des tranches intitulées « général ».

3. Interconnections des différentes tranches.

4. Ajout d'un label sur les liens.

Le résultat de cette phase donne le schéma suivant pour notre problème :

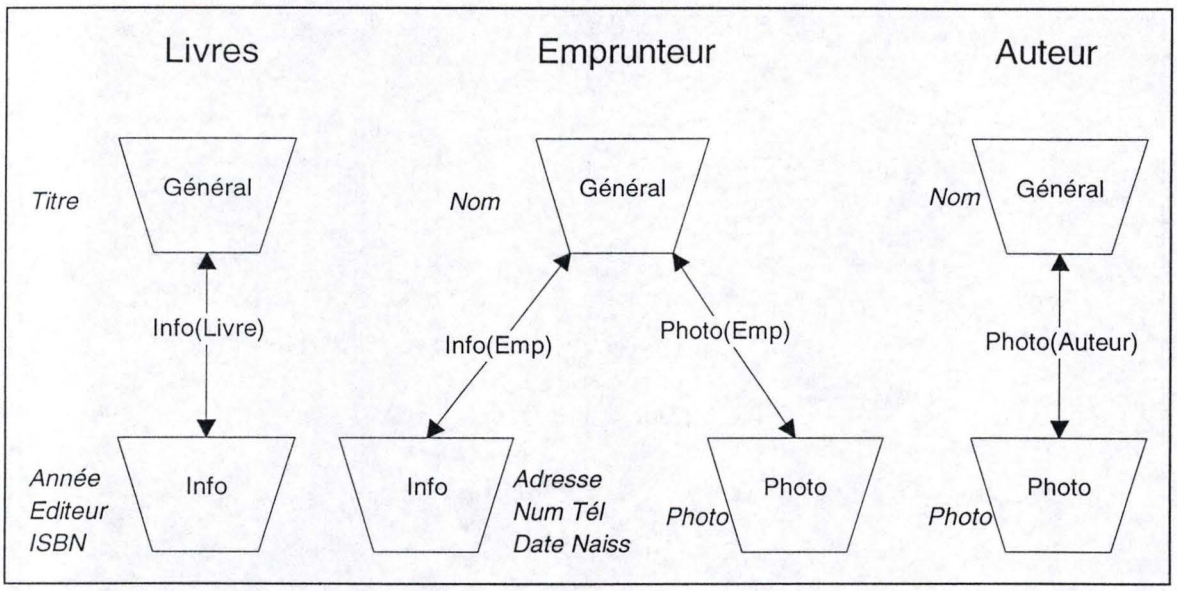


Figure 2-3 : Découpe de tranches

2.2.2.2 *Design de la navigation*

Les étapes à réaliser sont les suivantes:

1. Identification des informations et des relations accessibles.
2. Identification des groupements
3. Identification des structures d'accès.

Notons que les phases un et deux sont déterminées et se basent sur les pré-requis définis plus tôt et sont en partie traduits dans le schéma ER. Les règles à suivre pour la troisième phase sont celles-ci :

- une relation 1-1 est remplacée par un lien bidirectionnel,
- une relation 1-N est remplacée soit par un tour guidé, soit par un index (quand N est inférieur à 10), soit encore par un tour guidé indexé.

Le résultat produit à l'issue de cette étape est alors le suivant :

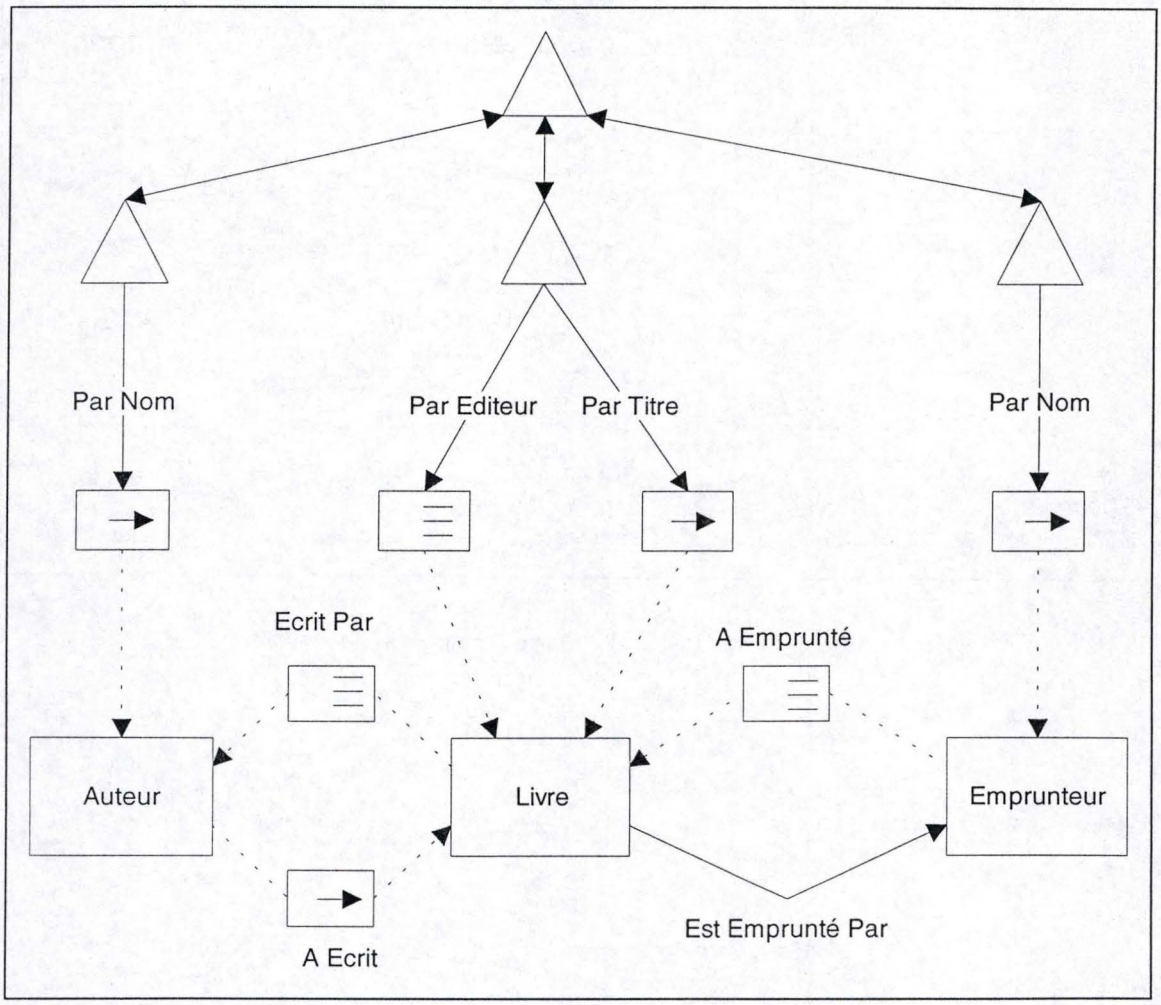


Figure 2-4 : Design navigationnel

Un éventuel reproche est que lors de la création d'un index, par exemple pour la recherche de livres, on est obligé de réaliser celle-ci par un seul attribut. Par exemple on sera obligé de rechercher un livre uniquement par son Titre ou son Editeur. Et de fait, on est obligé de préciser celui-ci dans la modélisation. Il eût été pratique de pouvoir modéliser une recherche sur un groupe d'attributs ou par un « masque » plus général (par exemple avec l'utilisation de jokers) [20].

Un autre désavantage est que l'on ne peut modifier ou ajouter des instances d'informations. En effet, RMM ne supporte pas ce genre de modélisation. Remarquons toutefois que la méthodologie nous a permis d'obtenir une modélisation assez bien adaptée à notre problème.

2.2.2.3 Les étapes suivantes

Les étapes suivantes, plutôt orientées implémentation, consistent à transformer les schémas produits en documents de type choisi par rapport à l'environnement de programmation et d'utilisation. Par exemple, dans le cas du Web, il existe des règles qui transforment les différents liens (comme le tour guidé ou l'index) en un groupe de pages HTML reliées entre elles par des liens hypermédias. Mais la littérature peu étoffée à propos de ces phases plus techniques ne nous a pas permis d'aller plus loin dans notre application.

2.2.3 Application de la technique SHDT

Notons tout d'abord que cette méthodologie ne se base pas sur le modèle ERA. Mais le concepteur peut s'en inspirer fortement lors de la première phase consistant à identifier les informations présentées à l'utilisateur.

Ensuite, il faut ajouter des liens entre ces informations. Dans notre cas, pour utiliser au mieux les possibilités de SHDT, nous avons modélisé la recherche et la consultation. Cette technique offre plus de possibilités, dont par exemple, la création de pages statiques de présentation. Par contre, comme dans le cas de RMM, elle ne prend pas en compte la modification ou l'ajout d'informations. Le schéma résultant est le suivant :

[20] Par exemple, la recherche de tous les livres paru en 1997 et dont le numéro ISBN contient le groupe de caractères 'BE' pourrait se réaliser en complétant les champs de recherche Titre par une étoile, Année par 1997, Editeur par une étoile et enfin ISBN par %BN%. Notons que cet exemple est fortement orienté implémentation.

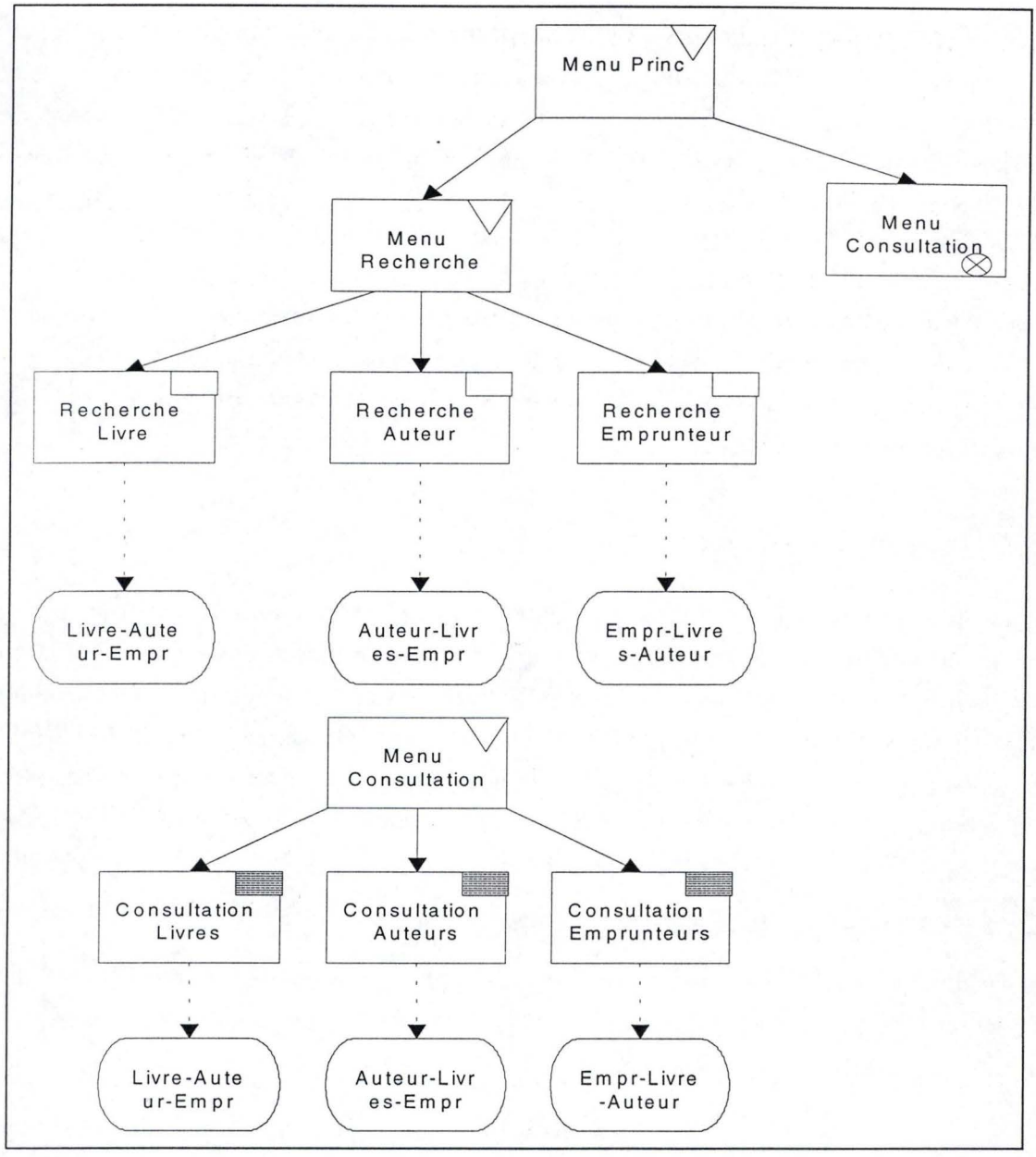


Figure 2-5 : Design avec SHDT

Les phases de design sont simplifiées par l'emploi de l'outil Web-Designer, implémentant SHDT. Une fois le schéma dessiné et les propriétés spécifiées, le programme génère les pages HTML et les scripts CGI adaptés permettant de retrouver les informations désirées dans la base de données.

2.3 Discussion générale

L'objectif de cette partie est avant tout d'apporter un autre éclairage concernant l'étude qui a été faite au chapitre 1 sur les méthodologies existantes (cfr. section 1.3) et les outils de conception assistée de pages HTML (cfr. section 1.2), dans le cas particulier de notre première classe de problèmes. Les méthodologies étudiées sont rapportées au cas étudié à la première sous-section, et une discussion sur les outils de générations assistée de pages HTML est développée à la sous-section suivante. La dernière sous-section, quant à elle, concerne quelques éléments de réflexion vis-à-vis de l'implémentation possible d'un outil de génération automatique de formulaires, intégrant les différentes phases de développement.

2.3.1 Méthodologies

Le design d'applications hypermédias doit tenir compte de différents points de vue tels que la structuration de l'information, le design navigationnel et le design de l'interface. Le design de bases de données est très proche de celui d'hypermédias. En effet, beaucoup de méthodologies formelles de design d'hypermédias se basent sur des modèles bien définis de design de bases de données comme le modèle ERA. Cette approche est très pratique dans le cas d'informations hautement structurées et volatiles telles que des catalogues de produits, fournissant alors des applications composées de pages dynamiques.

D'autre part, certaines applications ne traitent que des informations statiques telles que des encyclopédies ou autres regroupements de documents.

Enfin, il arrive souvent de trouver une combinaison de ces deux situations. Il n'est pas rare en effet de trouver, sur le Web, des sites proposant des informations statiques (pages d'explications générales) et dynamiques (recherche dans un annuaire) sur une même société par exemple.

Les différentes méthodologies étudiées couvrent la création d'hypermédias appartenant aux classes citées ci-dessus. Essayons de les classifier selon ces critères :

		Volatilité des informations	
		Basse	Haute
Structuration des informations	Haute	RMM (peu) SHDT (Application «kiosque»)	RMM SHDT EORM OOHDM SOHDM (Catalogue de produits, interface DBMS)
	Basse	SHDT (Travail Littéraire)	SHDT (Service de nouvelles Multimédias)

Tableau 2-1 : Classification des méthodologies étudiées

On remarque par exemple dans ce tableau, que si la structure des informations est haute et la volatilité basse, RMM est peu appropriée tandis que SHDT conviendrait.

En fait, comme on peut le constater, les méthodologies présentées, SHDT mis à part, se basent soit sur un modèle Entié-Relation, soit un modèle orienté objet de l'information. Ceci limite le champ d'action des applications à des informations plutôt structurées. SHDT sort un peu du lot en proposant de formaliser des informations moins structurées comme par exemple du texte brut.

Comme on le voit, aucune méthodologie n'est réellement polyvalente, par exemple, toutes ne se basent pas sur le schéma ERA, certaines fournissent un modèle navigationnel plus évolué que d'autres et il faut aussi remarquer que toutes les méthodologies ne sont pas applicables car les informations à leur propos sont difficiles à trouver.

RMM et SHDT ont la même lacune : la modélisation navigationnelle concernant d'une éventuelle mise à jour d'informations via l'application. De plus, les index sont un peu limités dans RMM (un seul attribut comme critère). La simplicité de SHDT rend la création de sites hautement interactifs et très riches assez difficile, vu le petit nombre d'objets disponibles ; de plus, seul l'aspect Web est pris en compte dans cette méthodologie (de part l'implémentation via Web Designer).

Enfin, il faut remarquer que l'exemple étudié est relativement simple. Il est certain que pour un problème dont le schéma conceptuel disposerait de plus de cinq entités, le choix de la navigation ne serait pas aisé.

2.3.2 Outils

Abordons maintenant l'apport que peuvent avoir divers outils pour les méthodologies abordées et les différentes phases qu'elles comportent.

Nous avons déjà remarqué que la technique SHDT a sa propre implémentation portant le nom de Web Designer. Cet outil est en cours de développement à l'Université de Vienne ; il se base sur la méthodologie SHDT (cfr. section 1.3.3), allant du schéma conceptuel à la génération de formulaires (il permet même d'individualiser les formulaires par le choix de feuilles de style et de gérer des contraintes quant à l'accessibilité de certains champs). Il est composé d'un éditeur supportant le modèle de la méthodologie et d'un générateur produisant les pages HTML et les scripts CGI. Notons que le développement de cet outil se poursuit parallèlement au développement et à l'amélioration de SHDT.

De même, mais d'une façon moins intégrée, l'outil RMCase (cfr. sous-section 1.3.2.3)supporte le design d'applications Web hypermédias utilisant le modèle de données de RMM, à savoir RMDM. Cet outil couvre bien les trois étapes de design (schéma ER, découpe en tranches et schéma navigationnel). Il permet enfin de déterminer un aspect commun pour les pages HTML qui seront générées. A notre connaissance, cette partie n'est pas encore entièrement développée.

D'une façon plus générale, un outil de génération et de traitement de schémas ERA tel que DB-Main conviendrait parfaitement à la réalisation du design de ce schéma. De même, la réédition du formulaire généré automatiquement peut être réalisée par des outils de gestion de sites Web tels que FrontPage 97 (cfr. sous-section 1.2.1.1). Un outil comme WebMania (cfr. sous-section 1.2.1.2)peut servir d'appoint pour de petites applications limitées à des formulaires très simples ne reposant pas sur un schéma conceptuel (par exemple une inscription en ligne d'une petite association de quartier).

Cependant, d'autres produits, beaucoup plus complets mais aussi beaucoup plus lourds à gérer, existent. L'un d'eux est fourni par Oracle (Designer2000, cfr. sous-section 1.2.2.1). Il s'appuie sur une base de données dont le schéma conceptuel a été dessiné dans ce même programme, non seulement pour générer la base de données physique et la création des requêtes afin d'accéder aux informations de cette base, mais aussi pour proposer un format de sortie HTML/CGI afin d'offrir une présentation sur le Web. La gestion de la navigation est, dans certains produits, davantage réalisée par programmation directe. Même si cela implique une connaissance plus fine d'un langage, l'avantage principal est de gérer le formulaire de manière adéquate, mais ceci au détriment de la facilité d'utilisation.

met quand à lui de créer des bases de données orientées objet. Il n'est pas permis directement un schéma ER mais, de part ses fonctionnalités de structure ...nique de classes ainsi que des possibilités d'héritage et de liens entre classes, il est tout à fait possible d'envisager une version « EKTOS » du schéma ER établi lors de la modélisation du problème. Chaque instance de classe (correspondant à une entité dans les schéma ER) est vue sous la forme d'une fenêtre dynamique. Les capacités d'affichage pour le Web sont mises en œuvre via l'option d'EKTOS permettant de générer des pages HTML à la demande. (cfr. sous-section 1.2.2.2).

Toutefois, en changeant nos exigences, il est possible de trouver certains programmes intéressants. Même si ceux-ci s'avèrent peu nombreux, il n'empêche que leur existence rend quasiment obsolète la résolution de ce problème initial. Ainsi, des outils de programmation tels que dbANYWHERE et Visual Café Pro permettent une très bonne gestion Web de formulaires créés à partir d'une base de données. Cependant, ces deux outils ne passent pas par l'intermédiaire du langage HTML mais par Java, afin d'afficher les informations sur le navigateur Web ; ils s'apparentent plus à de véritables environnements de travail pour l'utilisation d'un langage de programmation. Ces applications reposent alors sur une base de données dont le pilote ODBC* est employé pour la communication avec celle-ci.

2.3.3 Eléments de réflexion

Nous présentons ici quelques éléments de réflexion concernant la manière dont pourrait fonctionner un programme générant automatiquement ou de façon assistée des pages HTML à partir d'un schéma ERA. Ces éléments s'occupent de la partie technique de l'éventuel programme et non de l'aspect méthodologique. La première approche qui s'est présentée à nous face à ce problème de conception de formulaires Web était une approche centralisée sur la représentation interne du problème, à savoir le schéma ERA caractéristique de la situation.

Dans cette première analyse, trois phases se dégagent. La première consiste en la création du schéma ERA (celui-ci constituant la représentation interne du problème), la deuxième, en la génération du formulaire HTML (ceci étant la représentation externe du problème) et des différents scripts nécessaires pour l'interaction avec la base de données physique et la troisième, la correction éventuelle du formulaire ainsi généré grâce à un outil d'édition de pages HTML.

Dans notre esprit, la première étape doit se terminer par la création de la base de d relationnelle physique ; la génération automatique se baserait soit sur un langage permettant de travailler directement sur le schéma ERA, soit par une traduction du code ayant servi à créer la base de données (SQL ou non). En plus de cette génération automatique, une certaine interaction avec l'utilisateur serait indispensable dès cette deuxième phase : cette interaction se justifie étant donné que dans bien des cas l'utilisateur voudra personnaliser l'aspect final de ses pages (il pourrait également peaufiner la navigation au sein du formulaire).

Il conviendrait aussi, afin d'avoir un formulaire vraiment efficace, de gérer des contraintes sur l'accessibilité de certains champs. En effet, lorsque l'on génère des formulaires permettant d'interagir avec la base de données, il se peut qu'un champ soit restreint, c'est-à-dire que son accès soit réservé à seulement quelques utilisateurs au sein d'un groupe par exemple. On pourrait définir des niveaux d'accès aux informations suivant l'URL chargeant le formulaire, ce qui permettrait d'interdire et d'autoriser l'accès à certains champs pour certaines URL, ou bien, plus simplement, demander à l'utilisateur d'introduire un mot de passe quand il désire consulter ou modifier une information.

Un des aspects importants de la génération de formulaires HTML consiste, comme nous l'avons déjà souligné, en la présentation de ces documents générés. Il faut remarquer également que le contenu sémantique d'un formulaire est généralement très différent d'une utilisation à l'autre. De plus, les personnes ciblées (autrement dit les utilisateurs finaux potentiels, les personnes qui utiliseront leur navigateur afin de consulter la base de données part l'intermédiaire du formulaire généré) seront souvent d'horizons différents, il pourra s'agir d'enfants, d'adultes ou de personnes se trouvant dans une certaine catégorie socioprofessionnelle.

C'est pourquoi il est intéressant de considérer des environnements de présentation divers afin d'interagir au mieux avec les personnes concernées. Cet aspect peut même s'avérer indispensable si l'on souhaite que l'utilisation des pages HTML générées par le programme soit efficace et facilement compréhensible par un maximum de personnes.

Les Cascading Style Sheets proposées par le consortium WWW sont une bonne solution à ce problème. En effet, il devient aisé de proposer un ensemble de présentations de pages HTML les plus diverses, le programme de génération fournirait donc en sortie un document HTML auquel il suffirait d'appliquer un certain style choisi auparavant. Les styles envisagés pourraient être un style « moderne », un style « professionnel » ou encore un style « classique ».

Enfin, la génération automatique des formulaires HTML, aussi bien réalisée qu'elle puisse l'être, ne saurait dans tous les cas être conforme aux goûts de tous les utilisateurs. C'est pourquoi il serait intéressant de prévoir un outil permettant de retravailler les pages HTML générées. Cet outil devrait être capable de gérer les formulaires HTML mais aussi la retouche et la gestion d'images, la gestion de Frames, etc. Il devrait être également assez puissant de manière à ce que l'utilisateur ait un contrôle visuel du document (autrement dit il doit supporter une édition WYSIWYG).

2.4 Conclusion

Suite à l'analyse de certains outils et/ou méthodologies, nous pouvons constater que diverses solutions ou parties de solutions recouvrent déjà le domaine abordé dans ce chapitre. Différents outils supportant ou non certaines méthodologies existent.

Notre première classe de problèmes se ramène alors à une intégration de ces outils pré-existants ou au choix de ceux-ci dans leur entièreté ou en partie. De même, l'application d'une partie ou de l'entièreté d'une méthodologie existante peut la résoudre. Ainsi, les représentations interne, conceptuelle et externe du problème seraient à même de trouver un correspondant méthodologique et technique.

Toutefois, si cette classe de problèmes se trouve résolue, il ne faut pas perdre de vue qu'elle reste assez étroite, se limitant aux applications de gestion simple dont le schéma de base est relativement élémentaire. Les attributs sont de type simple, les applications n'affichant bien souvent que du texte et l'interaction permise à l'utilisateur assez rudimentaire, la personne ne pouvant bien souvent que naviguer entre des pages ou effectuer des recherches simples.

Le chapitre trois concerne quant à lui l'étude d'un problème plus vaste et correspondant davantage aux possibilités d'interaction que l'utilisateur est en droit d'attendre d'une application de gestion plus élaborée. De plus, nous souhaitons étendre le contexte d'application en augmentant l'interaction possible entre l'utilisateur et l'interface (par exemple en introduisant des techniques d'interaction vues au premier chapitre) et en ajoutant des types d'attributs au schéma conceptuel pour rendre l'application plus « multimédia » (en proposant par exemple des images, du son ou des vidéos).

On remarque aussi qu'il est important de délimiter un domaine particulier quant à l'analyse du problème. En effet, la sémantique est très importante car celle-ci transparaît dans une représentation externe, la représentation interne étant, rappelons-le, le schéma conceptuel de la base de données associée. De plus, on peut noter que dans le problème étudié dans ce chapitre, la représentation externe ne présentait pas de réelle sémantique.

Un des objectifs du chapitre suivant est de montrer que la représentation externe contient une structure sous-jacente, que celle-ci est en liaison étroite avec la représentation interne, et ensuite de repérer ces liaisons. Soulignons l'aspect important de cette représentation externe du fait qu'elle constitue l'interface, le seul contact avec l'utilisateur final. L'emploi de nouvelles techniques d'interaction va se révéler intéressant.

Dans cette optique, nous pouvons nous demander si les méthodologies et/ou outils employés dans le problème initial sont, oui ou non, encore valables. Nous allons essayer de critiquer leur efficacité et d'en faire apparaître leurs limites et leurs lacunes.

Chapitre 3. Développement de formulaires multimédias interactifs

Ce troisième chapitre aborde une seconde classe de problèmes plus générale que la génération de formulaires simples étudié au chapitre deux. Après avoir donné l'analyse contextuelle de celle-ci dans la première section ainsi que ses objectifs, nous décrirons les principales différences existantes avec les problèmes du chapitre précédent. Nous aborderons ensuite le problème de la construction d'une requête SQL via notre Applet Java réalisée lors de notre stage de maîtrise et enfin, la dernière section nous permettra de conclure.

3.1 Contexte

Le contexte consiste, pour ce chapitre, en des applications multimédias utilisant des bases de données reposant sur un schéma conceptuel. Ce dernier utilise des types de données multimédias (images, sons, vidéos, ...) en plus des types simples déjà évoqués au chapitre précédent.

Ces applications sont également particulièrement destinées au monde du World Wide Web. Dans ce cadre peuvent être intégrées les applications permettant par exemple la recherche dans des catalogues et/ou index de produits de consommation, de personnes, de voitures ou de pièces détachées ; l'accomplissement d'achats et dans un cadre plus général le commerce électronique peut être aussi envisagé. Est aussi concernée la consultation d'ouvrages appartenant à une bibliothèque, d'œuvres d'art, de galeries ou de photos.

3.1.1 Analyse de la tâche

3.1.1.1 Pré-requis

La tâche étant simple, les pré-requis sont à nouveaux *minimaux*.

3.1.1.2 Productivité

Faible. Il s'agit d'une utilisation occasionnelle. Dans notre exemple, l'utilisateur fera une recherche d'un ou de quelques produits à acheter, sans plus. Une fois son but atteint, il ne réutilisera peut-être le système qu'après un laps de temps assez long.

3.1.1.3 Environnement objectif

Existant. Il existe par exemple des catalogues de produits textiles (cfr. « La Redoute » ou les « 3 Suisses »).

3.1.1.4 Reproductibilité de l'environnement

Praticable. En effet, le système est la transposition informatique d'un catalogue ou index existant dans le monde réel.

3.1.1.5 Structuration de la tâche

Modérée. La structuration n'est plus élevée comme dans le cadre du chapitre précédent. En effet, la marge de liberté laissée à l'utilisateur est faible mais dans ce cas, des possibilités nouvelles sont disponibles et l'utilisateur a plus de choix.

3.1.1.6 Importance de la tâche

Faible. Dans la plupart des cas, la tâche n'est pas vitale pour l'utilisateur.

3.1.1.7 Complexité de la tâche

La tâche peut tout de même être d'une complexité *modérée*. En effet, elle n'est pas aussi simple que l'impression d'un document mais pas aussi complexe que la gestion d'une centrale nucléaire.

3.1.1.8 Première dérivation

Suite à l'analyse des paramètres relatifs à la tâche analysée, on peut de nouveau dériver un style d'interaction à l'aide des tables de la méthodologie Trident. Ce premier groupe de valeurs ne nous permet pas de déterminer exactement un style d'interaction. Cependant les styles questions/réponses, remplissage de formulaires et manipulation directe s'en rapprochent.

3.1.2 Type d'utilisateur

L'utilisateur est à nouveau « monsieur tout le monde », ayant tout de même accès à un ordinateur de type PC (personnel ou via une unité de service), désirant par exemple acheter un produit via Internet.

3.1.2.1 Expérience de la tâche

Généralement *moyenne*. L'utilisateur doit connaître une certaine base quant au domaine d'application. Par exemple, la personne doit connaître l'existence et la signification de l'échelle des tailles de vêtements ou des pointures de chaussures.

3.1.2.2 Expérience du système

Egalement *moyenne*. Les rudiments de l'utilisation du Web. Cependant, une solution graphique devrait être plus simple, plus intuitive à utiliser qu'une application se basant uniquement sur des éléments de type texte, comme pour la première classe de problèmes abordée au chapitre deux.

3.1.2.3 Motivation

Modérée. La motivation dépend de l'utilité de la recherche effectuée par l'utilisateur.

3.1.2.4 Expérience d'un moyen d'interaction complexe

Elémentaire. Cette expérience n'est pas requise pour le type d'application envisagé.

3.1.2.5 Seconde dérivation

Les tables nous proposent les styles d'interaction suivants : remplissage de formes et interaction iconique.

3.1.3 Description de l'environnement

L'environnement est le monde du World Wide Web et en particulier des formulaires multimédias décrits dans le langage HTML, utilisant des techniques d'interaction plus complexes tels que les images cliquables ou mondes virtuels.

3.1.3.1 Type de traitement

Multitraitement. L'application s'exécutant sur une machine de type PC (compatible IBM, Macintosh ou autre), celle-ci n'est pas uniquement dédiée à la seule application concernée.

3.1.3.2 Capacité de traitement

Moyenne. En effet, l'utilisation de l'application peut s'insérer dans un travail plus large. Dans notre cas, il peut être complémentaire à la création d'un document multimédia de présentation.

3.1.3.3 Troisième dérivation

Cette troisième analyse nous propose les styles d'interaction : remplissage de formulaires ou sélection de menu.

3.1.4 Attributs de dialogue

3.1.4.1 Contrôle du dialogue

Interne. En effet, pour l'utilisateur, l'introduction de valeurs qui serviront à la recherche du produit peut se faire de différentes façons, c'est donc lui qui possède le contrôle.

3.1.4.2 Mode de dialogue

Asynchrone. L'utilisateur n'est pas contraint de suivre un ordonnancement prédéfini.

3.1.4.3 Mode de déclenchement

Manuel explicite. L'utilisateur, par utilisation d'objets interactifs, plus ou moins visibles selon les cas, essaiera d'obtenir l'objectif qu'il s'est fixé.

3.1.4.4 Métaphore

Mini-monde. En effet, cette seconde classe de problèmes se base plutôt sur des tâches graphiques représentant, par exemple, l'objet à acheter ou à rechercher.

3.1.4.5 Quatrième et dérivation finale

Enfin, cette quatrième dérivation est plutôt favorable à l'utilisation du style d'interaction intitulé manipulation directe.

3.1.5 Conclusion

L'objectif est la conception d'applications WEB multimédias qui répondent aux tâches spécifiées ci-dessus pour les utilisateurs correspondant aux descriptions qui précèdent.

Si l'on reprend les différents paramètres étudiés, on peut dériver un style d'interaction final pour les applications de notre seconde classe de problèmes. Notre choix se porte donc sur la combinaison des styles d'interaction intitulés « remplissage de formulaires » et « manipulation directe ». En effet, nous désirons orienter l'interaction du client vers une approche plus graphique, plus intuitive et plus simple. Mais le monde du commerce électronique nécessite tout de même l'utilisation d'un style d'interaction plus classique comme le remplissage de formulaires car le premier style choisi ne permet pas de couvrir tous les besoins dans les limites de simplicité fixées (par exemple l'introduction des caractéristiques du client comme son nom ou son adresse).

3.2 Constatations et réflexions

Abordons les différences entre cette deuxième classe de problèmes et la classe simple étudiée au second chapitre. Premièrement, les types de données utilisés ne sont plus les mêmes. En effet, nous les étendons dans une première sous-section pour inclure les types de données multimédias les plus courants. Pour pouvoir réaliser la transition entre le schéma conceptuel (vue interne pour le concepteur) et l'interface utilisateur (vue externe pour l'utilisateur), il est nécessaire de créer un nouveau type d'attribut pour le schéma ERA.

L'approche qui pourrait être adoptée pour la construction de l'application Web est également différente. Il ne s'agit pas ici d'une « simple » traduction d'un schéma ERA en des formulaires élémentaires composés comme nous l'avons vu (par exemple de champs d'édition ou de boîtes à cocher). Dans cette deuxième classe de problèmes, une certaine sémantique va être donnée à la vue externe, à l'interface.

Cette sémantique sera très liée au domaine d'application du problème à résoudre. En effet, l'interface doit être représentative de ce problème et doit permettre à l'utilisateur d'arriver au but qu'il s'est fixé d'une façon optimum (par exemple l'achat d'une voiture ou la recherche d'une œuvre d'art dans un catalogue). Ces considérations constituent la partie de cette seconde sous-section.

Pour terminer, nous expliquons une approche de recherche graphique dans des catalogues. Cette approche est basée sur la notion de « Queries Dynamiques » pour laquelle une recherche se réalise par raffinement successif des possibilités offertes.

3.2.1 Extension du schéma ERA et types Mimes utilisés

Afin que les applications générées soient agrémentées de fichiers multimédias, il est nécessaire de pouvoir représenter ceux-ci dans le schéma ERA de départ. Dans ce but, un nouveau type d'attribut est créé et spécifié ci-dessous.

Ce nouveau format s'inspire du type Mime défini dans le RFC 1521. Celui-ci permet de donner un « type » à une information transmise via le système de mail traditionnel ou le protocole HTTP. Au niveau implémentation, EKTOS, le logiciel de base de données orienté objet opère d'une façon similaire. Pour lui, tout attribut étant un objet, on peut lui associer toutes les caractéristiques nécessaires, dont le type réel de la donnée.

Par ajout d'une en-tête Mime, le type du document transmis (une image, un son, une vidéo) peut être facilement identifiable. Le type Mime connaît une grande extension car il est maintenant souvent utilisé via le World Wide Web. En effet, les serveurs HTTP ajoutent également une en-tête Mime aux fichiers transmis. Le navigateur recevant le fichier n'a plus qu'à réaliser une recherche dans ses tables de correspondance pour connaître la façon de traiter les données (affichage, appel a un programme externe, ...)

En étendant le schéma ERA via l'ajout du type Mime, nous facilitons de la même façon l'identification du type réel du fichier multimédia. Le nouveau type proposé est défini de la façon suivante :

Attribut de type Mime:

```
Valeur: URL du fichier
Propriété (obligatoire): type Mime (composé d'un string défini)
```

Du point de vue de l'implémentation, un attribut de type Mime sera traduit en un champ composé de deux sous-champs de types string dans la base de données. Le premier serait l'URL du fichier ; le second, le string définissant le type réel du fichier.

Cette façon de travailler permettrait dans le cas d'un générateur de fichiers HTML d'inclure le fichier multimédia en utilisant le Tag HTML adapté.

Les types retenus, et les plus traditionnels sont :

Type	Sous-type
image	jpeg
	gif
audio	basic
vidéo	mpeg
	quicktime

Tableau 3-1 : Types Mime retenus

NB: le string du type se compose de la façon suivante: `<type>/<sous-type>`, par exemple « `image/gif` ».

3.2.2 Rôle de la sémantique du domaine d'application - Lien entre les différentes représentations

Dans notre première classe de problèmes, la solution finale consistait en une « traduction » d'un schéma ERA en éléments d'édition HTML simples. Cette traduction peut être automatisée avec plus ou moins de réussite. Le passage de la représentation interne (par exemple un schéma ERA ou une découpe d'une base de données en vues) en une représentation externe (le formulaire visible et utilisable) via la représentation du concepteur (le fichier HTML servant à décrire les formulaires) peut donc se réaliser selon des règles précises car il existe une équivalence entre les modèles. Un attribut texte contenant le nom de l'auteur d'un livre devient un champ d'édition permettant de réaliser une recherche sur ce critère, un attribut image vient s'afficher sur la page Web, et ainsi de suite. Chaque équivalent externe (par exemple un champ d'édition ou une image) ayant sa représentation pour le concepteur (code HTML approprié), représentation provenant de la vue interne (schéma ERA, attribut d'une base de données, etc.).

Dans la classe de problèmes étudiée dans ce chapitre, si la liaison entre le code HTML (représentation du concepteur) et l'aspect de l'interface sont toujours présents, il n'en va pas de même pour ce qui est du passage entre la vue interne et externe. Il faut parvenir à représenter, graphiquement cette fois, une sémantique qui va fortement dépendre du domaine d'application concerné.

La difficulté réside également dans le fait que la technique de recherche qui compose le cœur de l'application repose sur la représentation graphique des objets à rechercher. Les caractéristiques de l'implémentation viennent alors jouer un rôle important et diminuer l'efficacité des méthodologies étudiées voyant alors leurs modèles de données devenir inutilisables. En effet, leurs primitives permettent encore de modéliser l'architecture générale de cette seconde classe de problèmes (navigation simple entre différentes pages). Mais dès qu'il s'agit d'obtenir un niveau d'abstraction supplémentaire concernant les caractéristiques propres à la tâche envisagée (recherche graphique se basant sur de nouveaux types de données et autres techniques d'interaction), les méthodologies montrent leurs limites.

Dans cette optique, nous décrivons dans la sous-section 3.2.3 une technique de recherche particulière appelée « queries dynamiques » et proposons dans la section 3.3 un exemple concret d'une application permettant la recherche de valeurs dans une base de données par construction graphique de requêtes SQL.

3.2.3 Recherche par « Queries Dynamiques »

Actuellement, lorsque l'on extrait des informations d'un réseau, dans beauc
l'opération est effectuée par l'intermédiaire d'un butineur Web. Les tecl
recherche sont principalement basées sur l'introduction de mots clés, comme nou
considéré dans le chapitre précédent.

Le rôle de l'interface s'avère capital dans ce genre de problèmes, tant pour l'aspect consultation que recherche. En effet, dans ce dernier cas, tandis que des langages tels que SQL sont maintenant standardisés et que des formulaires prêts à être remplis deviennent monnaie courante, des requêtes dynamiques permettent d'établir une recherche de plus en plus complexe tout en facilitant la tâche de l'utilisateur par l'emploi d'une interface plus visuelle. Il est évident que les langages d'interrogation tel que SQL sont appropriés dans beaucoup de situations mais une recherche par manipulation directe et affichage du résultat sous forme graphique a beaucoup d'avantages dans un grand nombre de situations (cfr infra. 3.3).

L'exemple d'une telle situation est celui d'un programme de recherche géographique utilisé afin de retrouver un ensemble de maisons à vendre dans un secteur donné. Un système permettant à l'utilisateur de visionner le nombre de maisons à vendre dans un périmètre changeant dynamiquement selon ses désirs, tous ceux-ci étant affichés graphiquement (la carte de la région par exemple, ainsi que le périmètre choisi sur cette carte), permettra à l'utilisateur une meilleure approche de sa recherche que dans le cas où il l'aurait effectuée à l'aide d'un langage d'interrogation. Le succès de ces interfaces à manipulation directe est bien évidemment dû à l'émergence de systèmes informatiques de plus en plus puissants.

Les recherches dynamiques offrent à l'utilisateur un contrôle interactif de paramètres de recherche visuelle générant des résultats provenant d'une base de données ; ces résultats peuvent être animés, sonores, visuels, ou de quelque type que ce soit. La technique de recherche dynamique se base principalement sur une mise à jour visuelle des informations demandées et cela au fur et à mesure que l'utilisateur précise sa recherche. Il est donc capable de connaître le nombre d'informations ignorées pour sa recherche et peut ainsi affiner celle-ci comme il le désire, ayant un feedback visuel automatique. Ceci donne lieu à des interfaces utilisateur plus compréhensibles, plus contrôlables et plus prévisibles. Elles peuvent posséder les caractéristiques suivantes :

- une représentation continue des objets et des actions intéressant l'utilisateur,
- des actions physiques (cliquer sur des boutons et non pas utiliser une syntaxe compliquée),
- des opérations incrémentales, rapides et réversibles affectant directement et visuellement l'objet intéressant l'utilisateur.

Comme conséquence directe, on remarquera une absence de messages d'erreurs (ou du moins un nombre très faible).

« The success of direct manipulation stems from the visibility of the objects of interest so that interface actions are close to high-level task domain. There is little need for the mental decomposition of tasks into multiple interface commands with a complex syntactic form. On the contrary, each action produces a comprehensible result in the task domain that is visible in the interface immediately. The closeness of the task domain to the interface domain reduces operator problem-solving load and stress. » [SHNEIDERMAN97].

3.3 Construction d'une requête SQL via une interface graphique

Ce paragraphe décrit une des Applets Java que nous avons réalisée dans le cadre de notre stage de maîtrise à l'ISI (Information Sciences Institute) de l'USC (University of Southern California). Les éléments essentiels caractérisant cette Applet sont présentés dans une première sous-section ; les aspects plus techniques (tant au niveau implémentation qu'au niveau interface) sont décrits à la sous-section suivante. Enfin, nous terminons en tirant les leçons de sa construction.

3.3.1 Cadre et finalité

Nous avons effectué notre stage de maîtrise à l'ISI dans la division travaillant sur le projet DASHER (Defense Acquisition Services for High performance Electronic CommeRce). Ce programme propose une infrastructure commune donnant accès à des services indépendants, interopérables et modulaires pour le commerce électronique. En soumettant une interface aux clients via le Web, l'application permet la recherche dans une base de données intégrée, des fournisseurs proposant différents types de pièces (phase de « Quoting »). Ensuite, une recherche plus fine permet d'obtenir les prix pour une pièce donnée chez divers fournisseurs (phase de « Sourcing »).

On peut facilement imaginer la taille considérable de la base de données regroupant tous les renseignements utiles sur les fournisseurs, clients et marchandises. Il était donc nécessaire d'implémenter un programme qui permettait, dans les phases de tests, de créer facilement des requêtes afin de vérifier, par exemple, la cohérence des données.

Le but final était de pouvoir afficher graphiquement la structure relationnelle de la base de données et de pouvoir, toujours de façon graphique, créer une requête SQL en cliquant sur différents attributs de différentes tables et en y imposant certaines contraintes.

La finalité de notre Applet Java se situe dans ce cadre. Elle permet à une personne non initiée à la syntaxe SQL de créer une requête simple sur l'ensemble des données et d'en obtenir le résultat soit via l'Applet, soit sous forme d'une table en version HTML dans le navigateur avec lequel s'exécute l'application.

3.3.2 Implémentation

3.3.2.1 Architecture

Notre implémentation est réalisée en Java, le langage de programmation orienté objet, indépendant de la plate-forme. Avec Java, il est possible de concevoir de petites applications, les Applets, qui peuvent être transmises, via Internet, à un utilisateur distant. Nous avons réalisé de nombreux tests afin de trouver la meilleure façon d'utiliser ce langage pour notre finalité. Exécuter une Applet du côté serveur était une solution mais celle-ci s'avère difficile et peu efficace (cela induit en effet une surcharge du serveur Web).

Ensuite, nous avons testé les nouvelles classes d'objets « Internet Fundation » proposées par Netscape, qui proposent une nouvelle approche au niveau de l'interface utilisateur. De nombreuses et nouvelles possibilités sont offertes comme l'implémention du « Drag & Drop », la création de fenêtres au cœur même de l'Applet (qui peuvent par exemple être minimisées, agrandies ou fermées). Mais nos tests ont montré que cette approche était encore, à ce moment, trop jeune pour être utilisée efficacement. Il s'agissait effectivement à l'époque d'une version bêta[21] sans grande documentation ni exemples. Nous avons donc décidé de dessiner l'interface à l'aide des classes d'objets traditionnelles de Java, à savoir les AWT*.

Nous avons également utilisé les classes jdbcKona de Weblogic pour l'accès à la base de données. En particulier, jdbcKona/T3 est une implémentation multi-tiers JDBC* (basée sur la spécification JDBC 1.0 de Javasoft) qui permet aux Applets et autres applications Java d'accéder à une base de données via un serveur intermédiaire, en l'occurrence le T3Serveur de Weblogic. Il est nécessaire de passer par une architecture multi-tiers représentée par la Figure 3-1.

En effet, pour des raisons de sécurité, une Applet Java ne peut communiquer directement avec une base de données, sauf si celle-ci se trouve sur la même machine que celle dont elle provient. Le serveur intermédiaire écrit également en Java est, quant à lui, une application Java (il ne s'exécute pas via un navigateur) et peut donc communiquer avec la base de données. Comme il s'exécute sur la même machine que le serveur HTTP, il peut transmettre des informations via l'Applet.

[21] Une version bêta est une version non définitive d'un programme servant de base pour une série de tests précédent la commercialisation du produit.

3Serveur et la base de données, nous avons utilisé le driver de Weblogic Oracle. Mais d'autres drivers existent comme le jdbcKona/Sybase, MSSQLServer, ou même jdbcKona/ODBC.

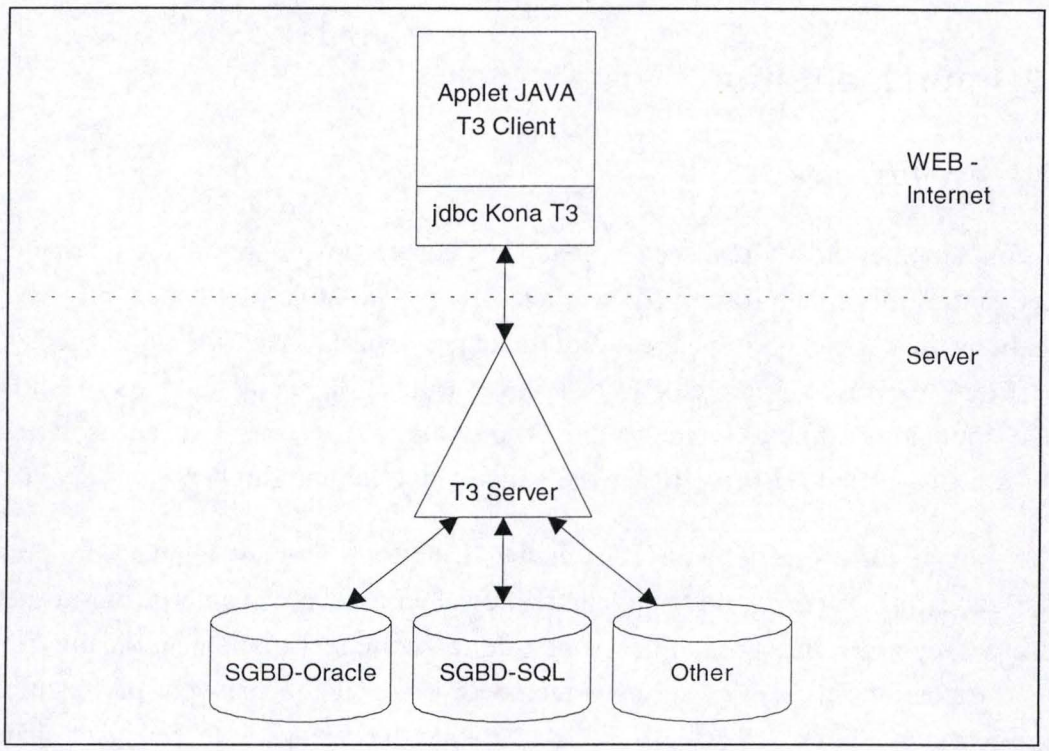


Figure 3-1 : Architecture trois tiers

3.3.2.2 Interface

Notre programme se compose tout d'abord d'un menu principal (Figure 3-2) composé de liens vers d'autres pages chargeant des Applets Java. Dans cette page de menu, deux choix sont proposés à l'utilisateur. Le premier lui permet d'exécuter une requête SQL (de type SELECT ... FROM ... WHERE ...) sur des tables d'une base de données, d'exécuter cette requête et de visualiser les résultats, et enfin de pouvoir sauver toutes les propriétés de cette requête (nom des tables choisies, choix des contraintes...) pour pouvoir ensuite, par le biais du deuxième lien, charger les informations précédemment enregistrées et de nouveau exécuter la requête.

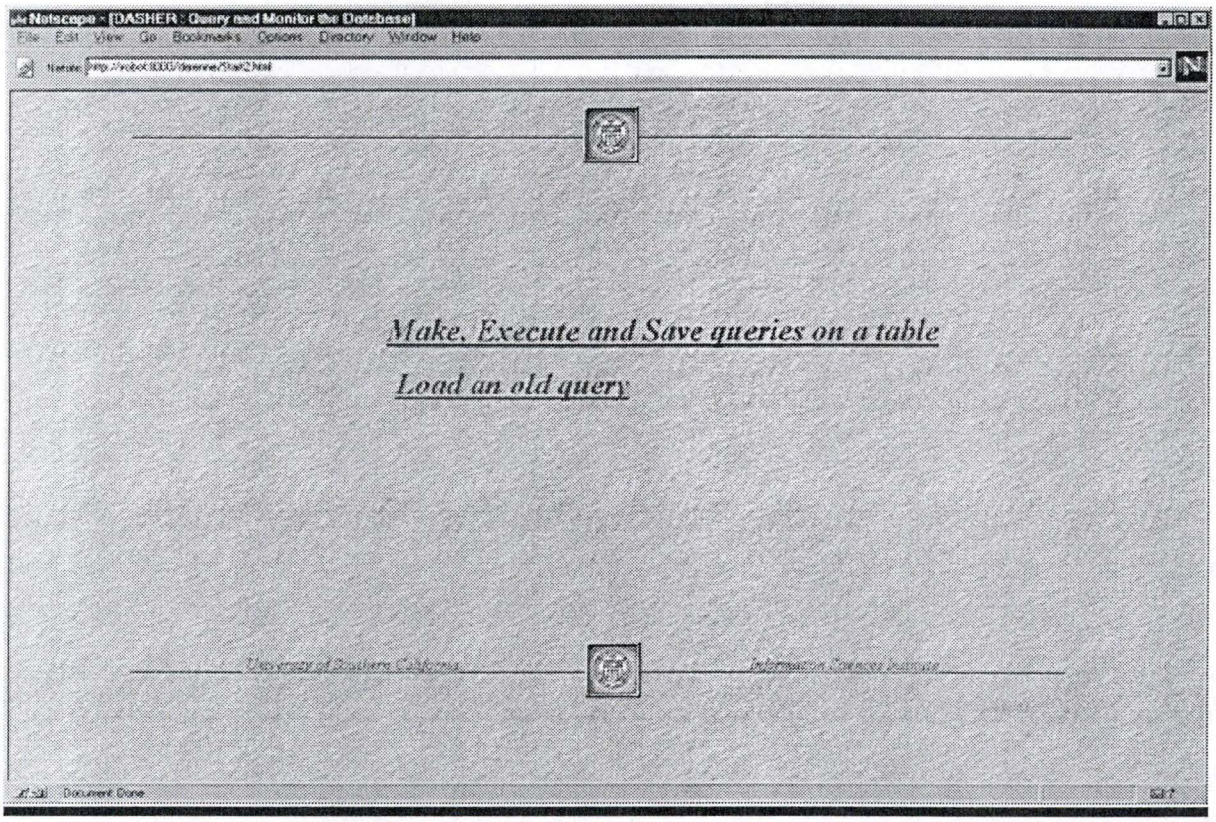


Figure 3-2 : Menu de l'application

Lorsque l'utilisateur choisit la première option du menu principal, il arrive à l'écran présenté comme à la Figure 3-3 (les numéros de 1 à 7 ont été rajoutés afin de faciliter l'explication de l'Applet). L'Applet est conçue de telle manière que l'affichage des tables existant dans la base de données soit réalisé dynamiquement. C'est-à-dire que l'affichage s'adapte automatiquement si, par exemple, une table est ajoutée entre-temps.

Tout d'abord, l'utilisateur doit attendre que toutes les tables apparaissent dans la première liste déroulante. Lorsque toutes les lignes sont affichées, il peut choisir une table (numéro 1). Les colonnes correspondant à cette table sont, quant à elles, affichées dans la deuxième liste déroulante. Pour imposer une ou plusieurs contraintes à une colonne ou l'afficher lors de la requête, il suffit à l'utilisateur de cliquer sur celle-ci (numéro 2) et de passer à la phase suivante correspondante. Ces premiers choix correspondent en fait à la partie FROM ... de la requête SQL.

La troisième étape (numéro 3) permet de définir si les éléments de la colonne sélectionnée doivent être affichés ou non lors de l'exécution de la requête, ou si l'on souhaite lui imposer une contrainte (ou les deux solutions). Pour ce faire, il suffit respectivement de cliquer sur le bouton « Show » ou sur le bouton « New ». Le premier bouton s'occupe de la partie SELECT ... de la requête, tandis que le premier de la partie WHERE ... qui est explicitée plus en détails dans le paragraphe suivant.

Si l'utilisateur a choisi « New », l'Applet lui permet d'accéder à l'édition de la contrainte portant sur la colonne voulue. Il lui suffit dès lors de choisir quel type de contrainte est apposé (quatre choix sont disponibles (numéro 4) par le biais de la liste déroulante ; ces choix sont <, >, =, between). Le premier champ d'édition, situé à gauche du numéro 4, rappelle la colonne sélectionnée ; quant au deuxième champ (numéro 5), il permet soit d'entrer une valeur, soit éventuellement une autre colonne pouvant correspondre à une autre table (dans ce cas on exécutera à nouveau les étapes 1 et 2 pour sélectionner éventuellement une nouvelle table et une colonne). Lorsqu'une contrainte est définie, il suffit alors de cliquer sur le bouton « Valid constraint » (numéro 6) afin de l'insérer dans la liste.

Il faut noter que l'étape 3 peut être répétée autant de fois qu'on le désire. De même, si une colonne à afficher ou une contrainte définie ne convient plus, l'utilisateur peut sélectionner le champ correspondant dans la liste déroulante qui convient et cliquer sur le bouton « Remove ».

L'étape numéro 7 permet à l'utilisateur d'exécuter la requête et de visualiser le résultat sous forme d'un tableau HTML, ou bien de sauver la requête (il devra alors entrer son nom d'utilisateur et le nom de sa requête ; remarquons qu'elle est sauvée dans une table spécialement créée au sein de la base de données), ou encore d'effacer et mettre à zéro tous les champs du formulaire.

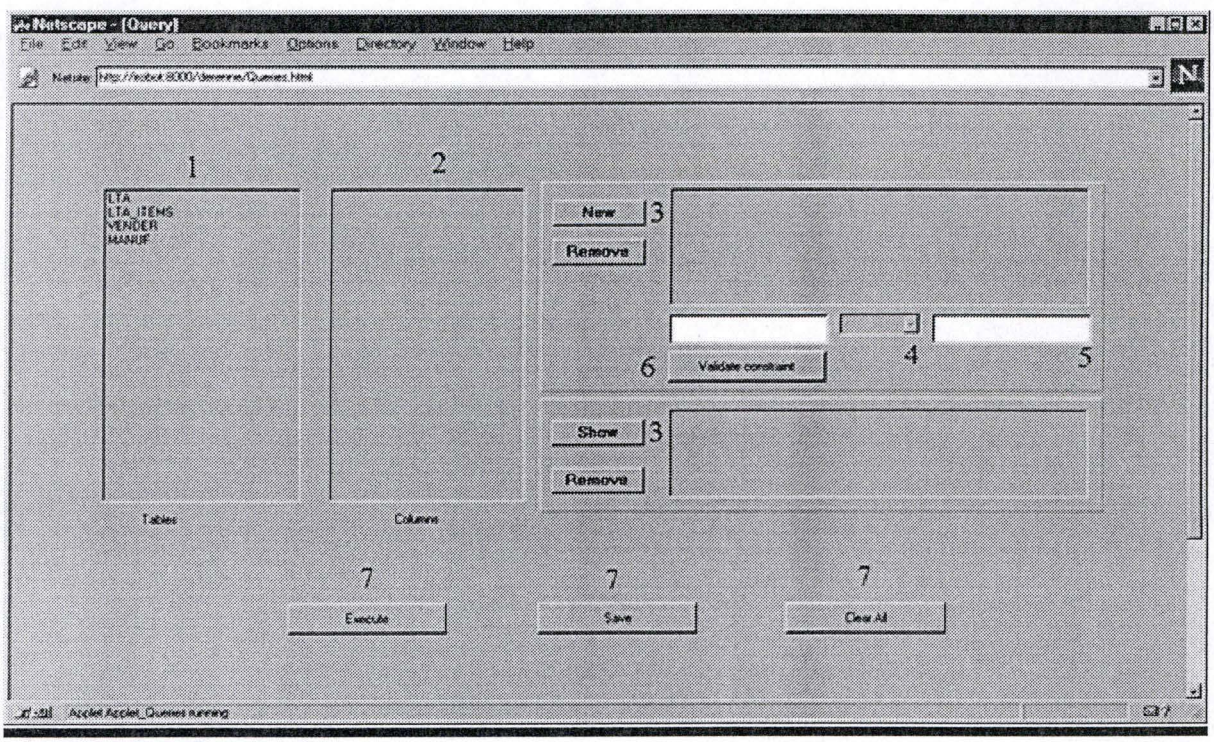


Figure 3-3 : Applet de création de requêtes SQL

3.3.3 Leçons tirées de la construction de cette Applet

Cette Applet Java permet la construction graphique d'une requête SQL par un utilisateur final qui ne connaît pas la syntaxe de ce langage. L'interface que nous avons construite est, en quelque sorte, un transfert de connaissances. En effet, nous pouvons conclure qu'il existe trois types de personnes concernées par ce problème :

- la personne qui connaît la syntaxe SQL et qui sait réaliser les requêtes,
- le concepteur qui va placer la connaissance de la première personne dans l'interface d'un programme en la formalisant,
- l'utilisateur final, qui ne connaît pas la syntaxe SQL, mais qui, graphiquement et en utilisant l'interface (c'est-à-dire la connaissance de la première personne), va créer une requête qui se révélera être, d'un point de vue implémentation, une requête SQL.

3.4 Conclusion

Cette deuxième classe de problèmes illustre assez clairement les besoins actuels des utilisateurs finaux en matière d'interface. Comme B. Schneiderman le signale [SHNEIDERMAN97], plus l'interface est convivial et dans ce cas adaptée à l'application de gestion considérée, plus la recherche dans les catalogues s'effectue de manière rapide et adéquate et plus l'utilisateur y trouve un confort d'utilisation (il pourra de nouveau visiter cet hyper-espace plus rapidement).

Comme nous l'avons vu au chapitre précédent, les méthodologies actuelles permettent de résoudre le problème initial, même si chacune d'entre elles présente des aspects positifs et négatifs. Etant donné que certaines de ces méthodologies sont implémentées dans des outils existants (par exemple RMCase cfr. sous-section 1.3.2.3) et que d'autres outils conviennent très bien en ce qui concerne l'implémentation physique (par exemple Oracle Designer 2000 cfr. sous-section 1.2.2.1), ce problème initial s'avère résolu. On est en droit de se demander s'il n'en est pas de même ici. En effet, les méthodologies existantes peuvent-elles résoudre le problème étendu ? Les outils existent-ils ?

la représentation interne du problème, les changements apportés au schéma
t l'apport de nouvelles caractéristiques multimédias, comme les types
nodologies, telles que RMM (cfr. section 1.3.2) et SHDT (cfr. section 1.3.3)
ant la génération d'applications hypermédias à partir de schémas conceptuels de bases de données, sont encore partiellement valables. Dans ce cas, il faut du moins de leur apporter de légères modifications afin de tenir compte des nouveaux types de données et de leurs implications dans l'interface qui sera présentée à l'utilisateur. On peut remarquer que les différentes phases des méthodologies concernant la conception du schéma ERA, étudiées auparavant, sont encore valables ici (même si la représentation externe du problème lui correspondra mieux). Toutefois l'aspect navigationnel des différentes méthodologies est en partie inadéquat.

En effet, ce qui change, c'est le passage à la génération de formulaires à partir de la modélisation conceptuelle du problème. Alors qu'auparavant, une sous-tâche était représentée par une interface spécifique, maintenant, et dans de plus en plus de cas, il est possible et même souhaitable de profiter des possibilités multimédias et des différents moyens d'interaction afin de fournir une interface graphique pouvant regrouper plusieurs sous-tâches et même une tâche entière (par exemple, une application de type commerce électronique peut être composée d'une tâche d'inscription du client, de la recherche de l'objet à acheter et de la confirmation de l'achat).

Dans cette classe étendue, la difficulté essentielle réside dans le fait qu'il est pratiquement impossible de générer automatiquement une interface de bonne qualité et hautement interactive. En effet, les difficultés qui y sont rencontrées dépendent fortement du domaine d'application. La génération que proposent les outils précités ne savent profiter pleinement des aspects fondamentaux de la classe de problèmes, et la génération, plutôt qu'automatique, doit se faire au cas par cas, en intégrant au mieux toute l'analyse préalable effectuée pour l'application de gestion choisie.

Dans le chapitre suivant, nous présentons un exemple correspondant à la classe de problèmes traitée dans ce chapitre. L'application de gestion choisie est la vente par correspondance de lunettes de soleil pour laquelle une interface graphique intégrant un meilleur dialogue avec l'utilisateur est présentée. Cette représentation externe sera la « vue » directe de la représentation interne correspondant au schéma ERA pour cette application de gestion.

Chapitre 4. Etude de cas : Achat interactif d'une paire de lunettes solaires par un client

Suite à l'étude générale de notre seconde classe de problèmes, nous élaborons un cas concret via cette étude de cas. Nous effectuons la description de la construction pas à pas ; les explications sont données en respectant l'ordre des étapes que nous avons suivies afin de réaliser l'application. Nous commençons par le choix et la description du problème. Ensuite, la récolte d'informations se rapportant au domaine d'application particulier est suivie par la modélisation de l'application. La construction proprement dite peut alors débuter avec le choix des outils et les différentes étapes de traduction et d'écriture. Cette étude concrète nous permettra dans le dernier chapitre de généraliser et de tirer les leçons et éléments susceptibles de constituer une méthodologie d'analyse, de modélisation et de construction d'applications de notre seconde classe de problèmes à savoir les applications interactives composées de formulaires multimédias pour le World Wide Web.

4.1 Choix et description du cas étudié

Le but de cet exercice pratique est de construire une application multimédia interactive dans le monde Web et dont la finalité est de permettre à des utilisateurs de choisir et de commander une paire de lunettes solaires dans un catalogue, ceci de façon graphique et interactive. D'une manière plus descriptive, l'utilisateur aura devant lui une interface qui lui permettra de choisir, caractéristique par caractéristique, les lunettes de ses rêves en sélectionnant les valeurs souhaitées pour les différents attributs possibles de la paire.

Cette interface, réalisée en HTML pour être visionnée via un navigateur Web, est donc la partie visible par l'utilisateur de l'application qui doit soutenir la résolution du problème. Les caractéristiques de l'interface sont les suivantes : l'image du visage d'une personne (éventuellement, selon les possibilités techniques, celle du client) est affichée à l'écran ; par superposition d'images, une paire de lunettes solaires se trouve sur celui-ci. Des zones sensibles sont définies et l'utilisateur pourra alors, en cliquant sur celles-ci, obtenir une liste des différentes possibilités de certaines caractéristiques. La métaphore du mini-monde est donc utilisée puisque c'est directement l'objet à acheter qui est représenté via l'interface et sert de moyen d'interaction pour le client. Nous reviendrons sur les détails pratiques de cette étape.

La construction de cette vue externe, à partir d'une base de données ou de façon plus générale d'un schéma conceptuel, dépend fortement de la sémantique liée à celui-ci, c'est-à-dire au domaine d'application du problème. Dans notre cas, nous construisons une modélisation représentative d'un ensemble de lunettes solaires. Les caractéristiques de ces lunettes (par exemple la couleur des verres, le modèle, etc.) sont également modélisées sous forme d'attributs. Dans notre première classe de problèmes, à savoir la construction d'applications composées de formulaires simples, le système était moins complexe : il suffisait de « traduire » ces attributs en équivalent HTML. Par exemple, du texte en champ d'édition ou encore une photo en emplacement d'image.

Il n'en est plus de même ici. Le but étant d'obtenir une représentation et une interface plus graphique pour la résolution du problème, il faut trouver les différents liens que l'ont peut obtenir entre celle-ci et les différentes caractéristiques des lunettes. Pour cela, nous avons tout d'abord essayé de mieux comprendre le mécanisme d'attribution et de choix de lunettes en questionnant des opticiens et en récoltant des informations dans des catalogues de grossistes.

4.2 Récoltes d'informations

Nos interventions auprès des opticiens ont pour but de récolter le plus d'informations possibles concernant le mécanisme d'attribution d'une paire de lunettes solaires pour un client donné. Dans cette optique, il était également nécessaire de connaître la composition d'une monture, par exemple les différentes pièces, caractéristiques, etc. De même, nous avons demandé quels étaient les facteurs plus ou moins importants jouant un rôle dans cette phase.

La consultation de catalogues obtenus auprès de grossistes nous a également permis d'avoir une idée des modèles existants, des types de montures plus ou moins modernes, ainsi que la diversité des pièces détachées existant sur le marché.

De cette récolte d'informations, nous avons pu conclure que le mécanisme fondamental du choix d'une paire de lunettes solaires pour une personne donnée et réellement appliqué chez les opticiens n'est pas facilement identifiable, et la mise en place de règles précises le décrivant est quasiment impossible vu le caractère subjectif très prononcé. Il existe effectivement des cours, lors des études d'optique, qui expliquent comment choisir le type général de lunettes pour une physionomie donnée (par exemple pour un visage plus rond ou plus allongé). Mais il ne s'agit que de règles générales qui ne fonctionnent que très rarement car ne prenant pas en compte de l'esprit de la personne concernée et d'autres caractéristiques plus ou moins subjectives.

Nous nous limitons alors à proposer, via l'application, une image de lunettes solaires permettant de choisir parmi les caractéristiques essentielles de celles-ci. La première paire proposée au client dépend tout de même de son âge et de son sexe. Nous définissons les autres caractéristiques dans l'étape de modélisation.

4.3 Choix des outils utilisés

Rappelons que notre but est de créer une application Web présentant une interface graphique et permettant à une personne de choisir une paire de lunettes, de la commander et de l'acheter. Citons les outils utilisés, leur production et la représentation sur laquelle ils travaillent.

Les outils nécessaires à la conception de cette application sont de différentes classes. Tout d'abord, un outil de conception de schéma ERA a été utilisé pour la modélisation du problème : il s'agit de DBMain, créé à l'Institut d'Informatique de Namur. Il génère également le schéma relationnel qui sert à créer la base de données, base gérée en l'occurrence par le logiciel Access de Microsoft. Typiquement, cet outil nous permet donc de modéliser le problème et d'en construire la vue interne via les modèles proposés. Les données de la base sont des éléments textes ou graphiques, par exemple les images des paires de lunettes. Ces dernières sont scannées et traitées avec le logiciel shareware* de traitement d'images Paint Shop Pro.

Un serveur Web est nécessaire à la diffusion des pages HTML : le programme Personnal Web Server[22] de Microsoft a été choisi pour sa simplicité de mise en place et sa gratuité. De même, un éditeur est utilisé pour créer les pages statiques et la base des pages dynamiques. Cette partie du travail n'étant pas très complexe, le programme Notepad livré avec Windows 95 a été utilisé. De plus, le langage HeiTML[23] , étendant le langage HTML, a été sélectionné afin de créer des pages HTML dynamiques. HeiTML propose des extensions comme la possibilité d'accéder à une base de données SQL via le standard ODBC, la définition de nouveaux Tags HTML pour la présentation de pages, ainsi que des possibilités de programmation comme les instructions de conditions, les boucles et les variables locales et globales. Ces différents outils nous permettent d'écrire l'application en elle-même dans le langage HTML étendu par HeiTML ; ils nous permettent de travailler sur la représentation du programmeur.

Notons que nous n'employons pas d'outils graphiques qui nous auraient permis de travailler directement sur l'interface utilisateur. Par exemple, en éditant graphiquement les pages HTML en plaçant par « Drag & Drop » les différents éléments de ces pages. La vue externe du problème, l'interface perçue par le client, est donc dérivée de la programmation même des pages HTML interprétées par un navigateur. Pour nos différents tests, nous avons employé le butineur Netscape Navigator. C'est celui-ci qui nous propose la représentation externe du problème.

[22] Le Personnal Web Server est un produit Microsoft : http://www.microsoft.com/infoserv
[23] HeiTML est un produit de la société allemande HEI : http://www.h-e-i.de

Outils	Type	Production
DBMain	Outil Case	Schéma ERA
DBMain	Outil Case	Schéma relationnel
Access	SGBD	Données issues de la base
Paint Shop Pro	Logiciel de traitement d'images	Images au format Gif
Notepad	Editeur de texte simple faisant office d'éditeur HTML	Création de pages HTML statiques et base des pages dynamiques en HeiTML
Microsoft Personnal Web Server	Serveur Web	Diffusion de pages HTML
Microsoft ODBC drivers 32 bits	Liaison ODBC	Accès à une base de données SQL
HeiTML	Extension du langage HTML	Scripts permettant l'accès à la base de données, traitement de celles-ci et création de pages HTML dynamiques.
Netscape Navigator	Navigateur Web	Interface utilisateur

Tableau 4-1 : Liste des outils utilisés

4.4 Modélisation

Trois notions essentielles sont nécessaires à la modélisation et la construction de notre application. La première est bien évidemment celle de lunettes, élément central du problème, choisi et acheté par le client. Ce même client constitue notre seconde notion, avec ses propres données. Enfin, pour permettre la mémorisation des différents choix des clients, la notion d'achat est modélisée juste avant de présenter la modélisation globale de l'application.

4.4.1 Lunettes

Objet central de notre future application, les lunettes possèdent différentes caractéristiques. Nous avons sélectionné les suivantes : le numéro de catalogue caractérisant le modèle, la couleur des verres, la couleur de la monture, le matériau de la monture, le sexe du client pour lequel celle-ci est destinée et enfin une photo représentant cette paire de lunettes, vue de face. Une paire de lunettes est identifiée par son numéro de catalogue et les caractéristiques du type de modèle, couleur des verres, de la monture et du matériau utilisé. Dans la vie réelle, le prix de la monture est une caractéristique importante. Cependant, il n'a pas été pris en compte dans notre exemple.

Toutes ces informations et mesures doivent faire partie de la base de données et être connues de l'application et donc se retrouver dans le schéma conceptuel. Il faut encore les représenter d'une façon ou d'une autre à l'utilisateur, afin que celui-ci puisse obtenir l'aspect final par choix successifs, dans la mesure du possible (tous les modèles correspondant aux différentes permutations des données n'existent pas sur le marché).

Lunettes
Numéro catalogue Type Couleur verres Couleur monture Matériau Fichier face Sexe
id: Numéro catalogue Type Couleur verres Couleur monture Matériau

Figure 4-1 : Modélisation de la notion de lunettes

4.4.2 Clients

Le client est l'utilisateur potentiel de l'application. Il est utile de le modéliser afin de connaître quelques données pertinentes à son sujet. En effet, le but étant l'achat d'une paire de lunettes, il est nécessaire de posséder le nom et prénom de la personne ainsi que son adresse, son numéro de carte de crédit, son adresse électronique. Des données secondaires comme son âge, son sexe pourront être utiles afin d'effectuer un premier « tri » de lunettes proposées. Un client est identifié par son nom, prénom et adresse électronique.

De même, on pourrait imaginer d'attribuer un mot de passe à une personne, ce qui lui permettrait de pouvoir revenir sur le site et par exemple de passer outre de la séance d'identification, ou de visionner les lunettes déjà achetées. Bien que ces possibilités concernent déjà un peu plus la phase d'implémentation, il est tout de même nécessaire de les modéliser.

Enfin, une image de la personne pourrait si possible être contenue dans la base de données. Si l'image provient de l'utilisateur, elle doit être à un format acceptable par l'application.

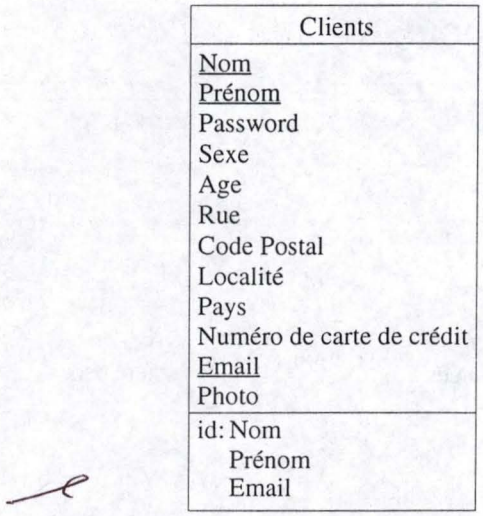


Figure 4-2 : Modélisation de la notion de clients

4.4.3 Achats

La modélisation de la notion d'achat nous permettra de mémoriser les différents choix déjà effectués par le client. Par exemple, lors d'une seconde visite, le client aura la possibilité de visionner les lunettes déjà achetées avant de passer de nouveau à la phase de sélection. Un client pourra passer plusieurs commandes concernant une paire de lunettes ; une paire, quant à elle, peut être commandée plusieurs fois.

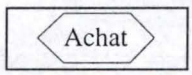


Figure 4-3 : Modélisation de la notion d'achats

4.4.4 Modélisation globale

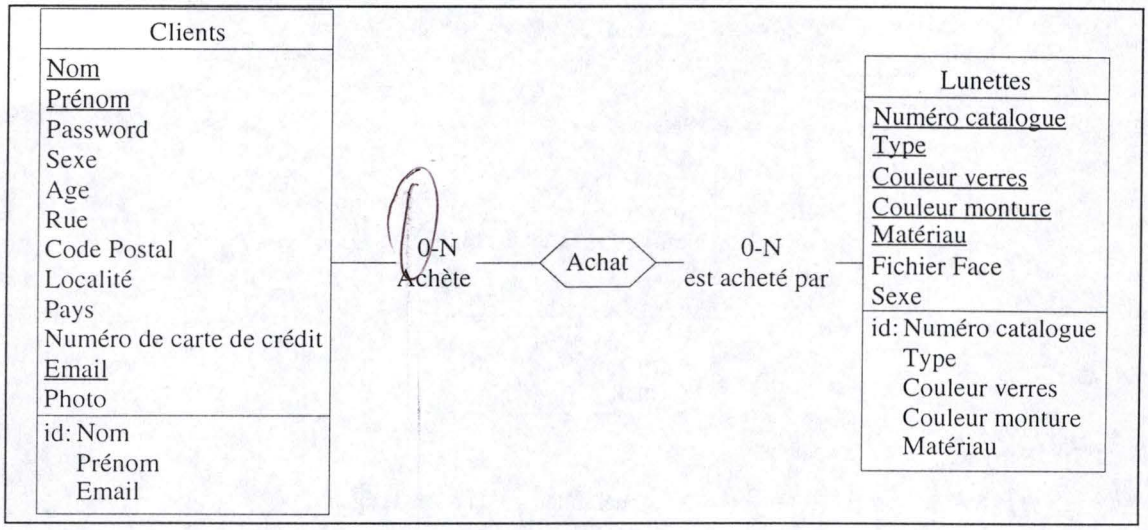


Figure 4-4 : Schéma ERA de notre problème

Un client est une personne qui s'est inscrite via l'application. Il peut avoir acheté zéro, une ou plusieurs paires de lunettes solaires. Une paire, à son tour, peut simplement faire partie du catalogue ou également être achetée par un ou plusieurs clients.

Précisons de suite que pour faciliter l'implémentation du problème, nous avons remplacé respectivement les identifiants des deux types d'entité par des attributs nommés « Ident » de type entier et identifiant donc une et une seule instance. Cette modification donne le schéma ERA suivant :

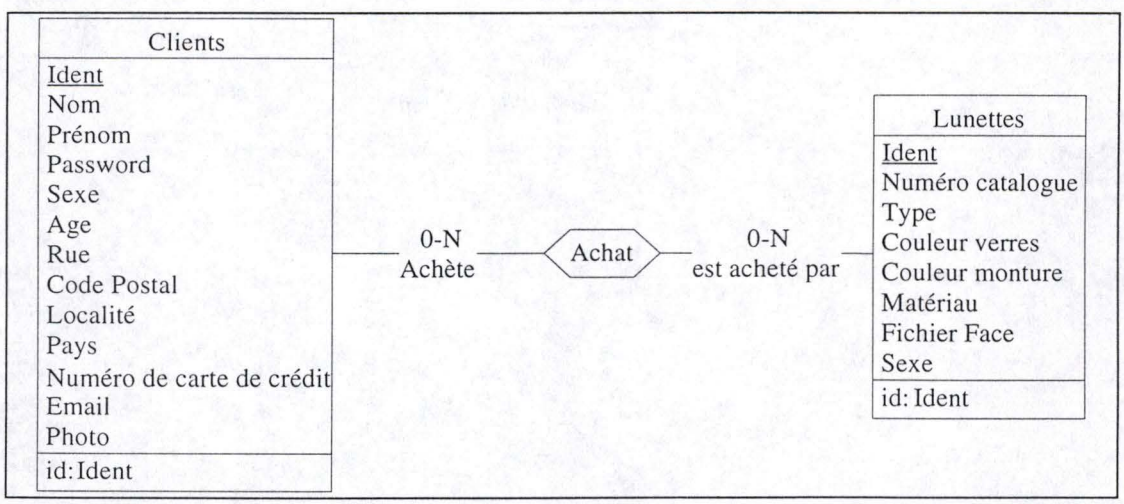


Figure 4-5 : Schéma ERA après transformation des identifiants

Donnons ici le schéma relationnel qui nous servira à l'implémentation du problème dérivé à l'aide de DBMain :

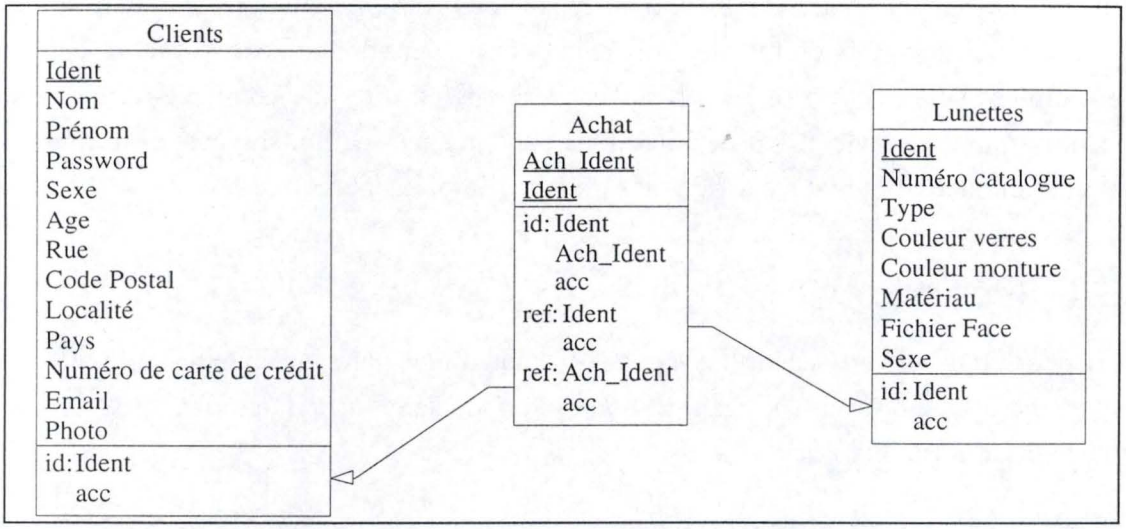


Figure 4-6 : Schéma relationnel dérivé

Ce qui donne pour la représentation Microsoft Access :

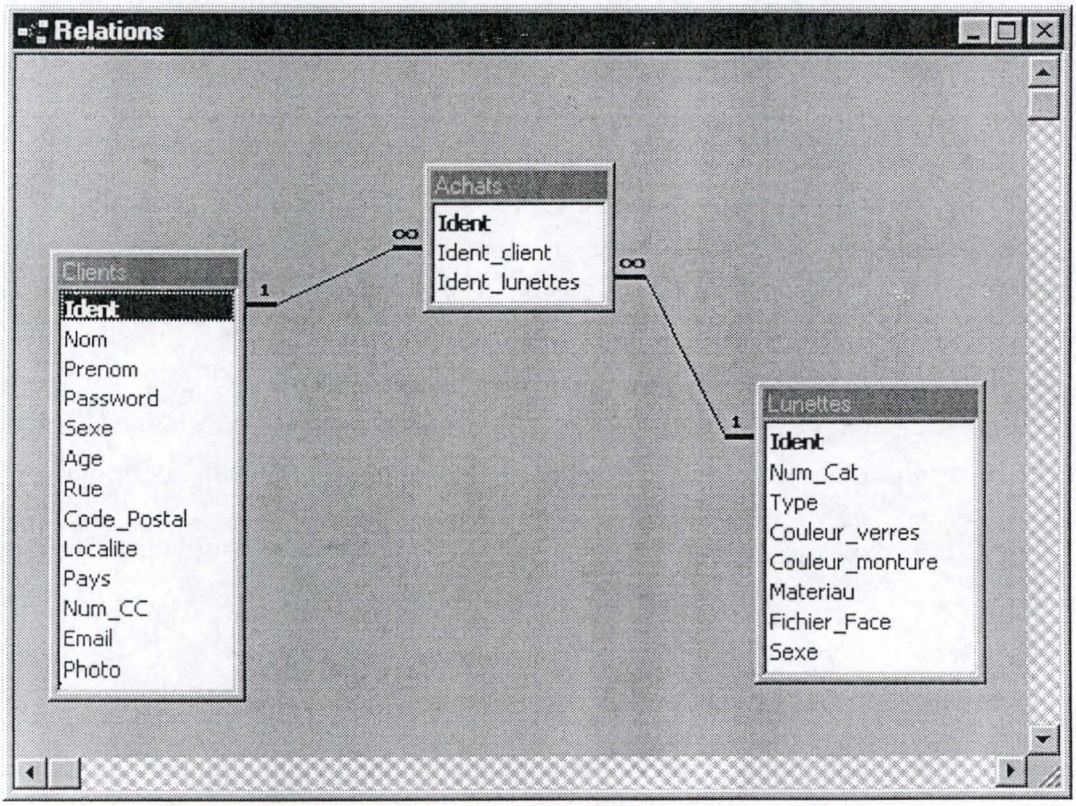


Figure 4-7 : Schéma relationnel version Access

nstruction

onstruire notre application, nous nous basons sur un éventuel scénario
on d'une paire de lunettes solaires pour un client donné ; ceci est représenté dans une première sous-section (4.5.1). Nous en dérivons ensuite une architecture, établissons des choix quant au style et but de l'interface dans la sous-section suivante et terminons la construction en expliquant les différentes phases d'implémentation (4.5.2).

4.5.1 Scénario

Nous décrivons ici, un scénario éventuel d'achat d'une paire de lunettes solaires. Cette description possible servira à organiser notre interface de la façon la plus intuitive et la plus efficace pour la finalité fixée.

Prenons le cas classique d'un client masculin âgé d'une trentaine d'années et désirant acquérir une paire de lunettes solaires pour partir en vacances d'été. N'ayant pas d'idée précise, il se rend chez son opticien habituel et lui demande de le conseiller dans son choix.

A ce moment, l'opticien dialogue un peu avec le client pour connaître le style général de sa future paire de lunettes. Il demande si son client les désire noires ou de couleur, avec de grosses montures ou celles-ci discrètes, le prix qu'il sera prêt à payer, etc. Avec ces premières données, il peut lui proposer une paire qui répond à ses premières exigences, une paire à monture fine en plastique bleu clair par exemple.

Le client, mécontent de la couleur, demande alors les teintes existantes pour ce modèle. Il remarque aussi une série de lunettes couleur aluminium. L'opticien lui propose une paire toujours à monture fine plastique, mais cette fois de couleur aluminium. Notre jeune homme apprécie maintenant la teinte, mais la matière n'est plus à son goût, elle dénote un peu avec la nouvelle couleur choisie. Le marchand peut ensuite lui proposer une série d'autres lunettes ayant les caractéristiques déjà choisies, c'est-à-dire des lunettes à monture fine de couleur aluminium. Il lui propose enfin des montures en titane ou aluminium.

Le choix final du client se portera finalement sur la paire en titane à monture fine couleur aluminium.

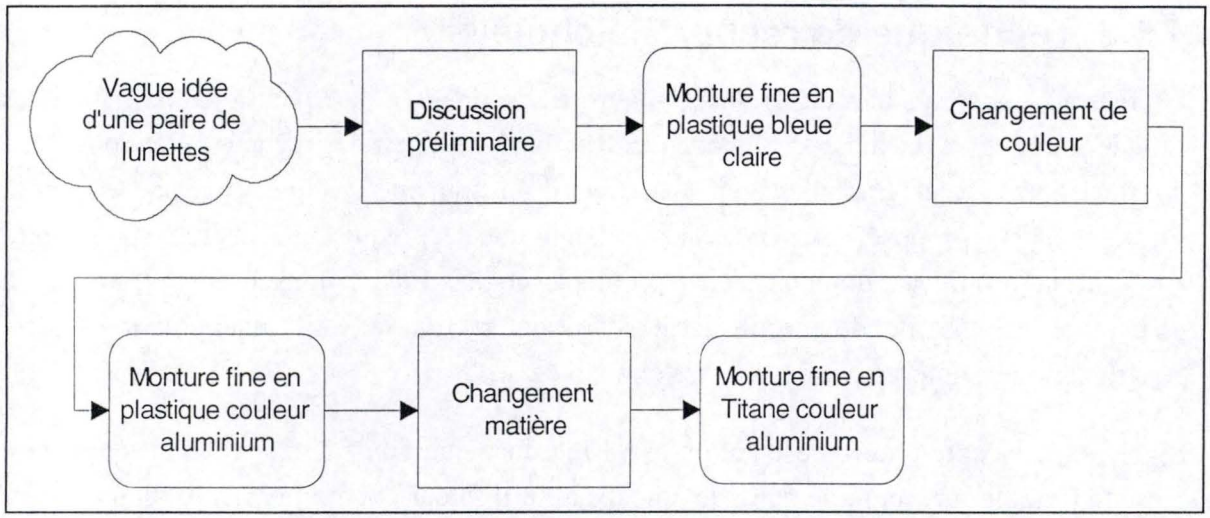


Figure 4-8 : Scénario possible pour le choix d'une paire de lunettes solaires

4.5.2 Architecture de l'application

L'application permettant le choix et l'achat d'une paire de lunettes solaires se découpe en quatre étapes distinctes représentées dans le schéma suivant :

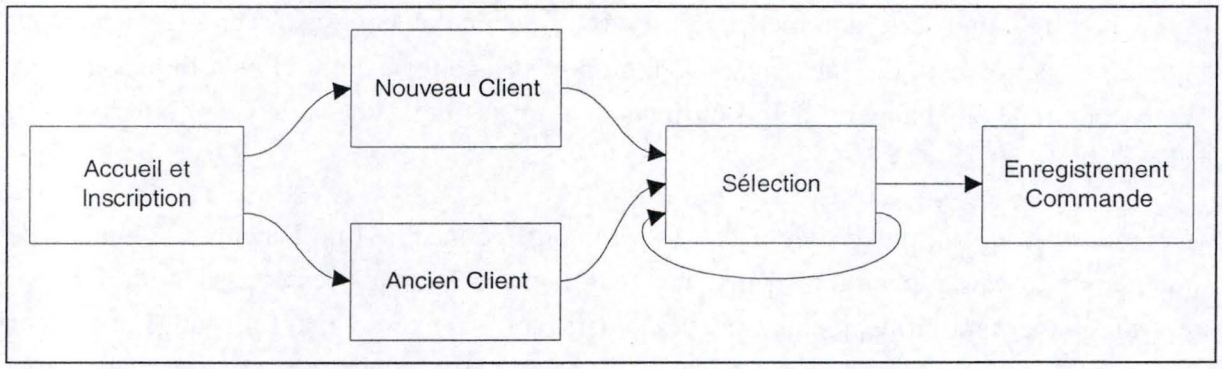


Figure 4-9 : Architecture globale de l'application

La première étape consiste à accueillir le client potentiel et à lui demander les informations le caractérisant. Si l'utilisateur est un ancien client, on lui propose éventuellement, dans la seconde phase, ses anciens achats. Sinon, le nouveau client reçoit un mot de passe qui lui permettra à l'avenir de passer outre cette première phase d'inscription.

La troisième phase, qui est le centre de l'application, consiste à proposer au client l'interface qui lui permettra de visualiser et choisir les caractéristiques concernant une paire de lunettes solaires qu'il est susceptible d'acheter. Une fois son choix confirmé, l'application propose un récapitulatif et l'enregistrement des données concernant l'achat.

4.5.3 Technique de recherche choisie

Intéressons-nous plus particulièrement à la troisième phase, concernant le choix même des caractéristiques de la paire de lunettes. En se basant sur nos exigences (création d'une interface intuitive et graphique, représentative d'un domaine d'application particulier) et sur le scénario proposé, nous pouvons élaborer une technique de recherche pour notre application, propre au domaine et au problème choisis. Nous nous basons donc sur la métaphore du mini-monde en utilisant le style d'interaction qu'est la manipulation directe et la technique d'interaction des images cliquables.

L'idée sous-jacente à la représentation et à l'interface que nous allons élaborer, consiste à présenter graphiquement une paire de lunettes à l'utilisateur et à lui permettre d'en changer les caractéristiques également affichées. Ceci lui est permis en cliquant sur une des zones sensibles de l'image. Chaque zone est liée à une caractéristique de l'objet et l'activation d'une de ces zones déclenche l'affichage des propositions d'autres paires ayant les mêmes attributs, sauf ce dernier. Le choix final se réalise donc par affinements successifs des différentes valeurs liées à l'objet de l'achat.

Prenons un exemple à titre indicatif. L'image actuelle est celle d'une paire de lunettes à monture métallique noire aux verres de couleur grise. Les mêmes verres définissent des zones cliquables de l'image et sont liés à la couleur de ces verres. Si l'utilisateur clique sur un des verres, une série de lunettes ayant des verres de couleurs différentes, mais gardant les autres caractéristiques inchangées sont proposées au client. Celui-ci peut faire son choix dans cette liste ou changer d'autres attributs de l'objet. Et ainsi de suite jusqu'à trouver son choix idéal.

Ce mécanisme, simple d'emploi et assez intuitif, donne à l'utilisateur une marge de manœuvre assez large, en lui permettant à tout moment de remplacer une valeur qu'il aurait éventuellement modifiée quelques étapes auparavant. Par exemple, ayant changé la couleur des verres, puis enfin le style de monture, il peut à nouveau changer la couleur des verres pour retrouver la teinte de départ (ou tout autre coloris disponible).

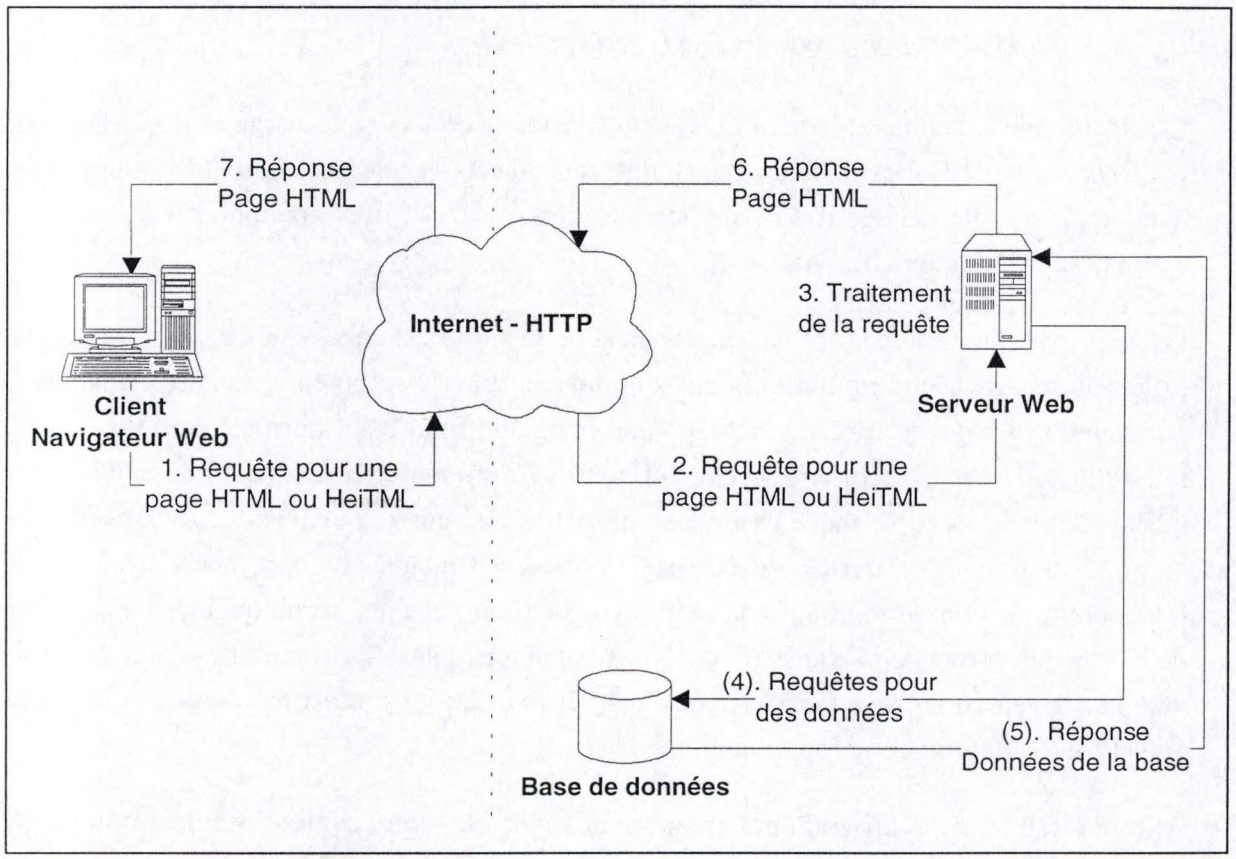
Un autre choix aurait pu se baser sur un mécanisme plus similaire à l'Applet Java réalisée dans le cadre de notre stage ou sur la notion de requêtes dynamiques, donc plutôt axé sur la construction itérative d'une requête dont le résultat s'affine au fur et à mesure des choix du client. Mais nous avons préféré la première manière de faire car, pour nous, elle se prête mieux à ce cas.

4.5.4 Phases de l'implémentation

Passons aux détails pratiques de l'implémentation de notre application. Cette étape se compose de l'installation, préparation des outils de conception et d'exécution, de l'écriture des pages HTLM et enfin de différents tests afin de parfaire l'ensemble.

4.5.4.1 Installation et fonctionnement des outils

Les différents outils nécessaires à la génération de pages HTML sont, comme déjà cités, un serveur Web, un programme reconnaissant le langage HeiTML étendant HTML, des drivers de connexion ODBC et une base de données. Examinons maintenant le fonctionnement à l'exécution de ces différents outils.



Un client, via son navigateur Web, envoie une requête (1) pour une page HTML ou HeiTML. Cette demande transite via le réseau jusqu'au serveur Web (2), qui la traite alors (3). Si l'objet de la requête est une simple page HTML, celle-ci est directement envoyée par le serveur jusqu'au client par le réseau (6 et 7). Par contre, s'il s'agit d'une page contenant des extensions HeiTML, le serveur fait appel à un interpréteur. Celui-ci lit la page et exécute éventuellement certaines actions comme la demande d'informations à une base de données (4) via les drivers ODBC. Recevant celles-ci (5), il peut écrire une page HTML qui sera envoyée au client via le réseau (6 et 7).

Remarquons qu'il existe cependant certaines contraintes au niveau de ces outils. En effet, le serveur Web doit reconnaître les requêtes pour les pages HeiTML. L'ordinateur doit, de plus, posséder les drivers ODBC et, la base de données employée, doit supporter les demande d'informations via ce standard. Dans notre cas, le « Personnal Web Server » de Microsoft était reconnu d'emblée à l'installation de HeiTML. De même, Access 95 possède la possibilité d'être accédée via ODBC.

4.5.4.2 Ecriture des pages HTML et HeiTML

Précisons tout de suite au lecteur qu'il peut trouver le code source complet des différentes pages en Annexe E. Les différents enchaînements des pages et leur contenu sont expliqués dans la section de descriptif de l'application (cfr. section 4.6). Nous donnons ici quelques éléments d'implémentation utilisés.

Le langage HTML ne permet pas encore la superposition d'images dans une même page[24]. Pour contrecarrer cette limitation, nous simulons la superposition en plaçant le visage de la personne en image de fond de la page. Une image de fond étant normalement répétée sur toute la page pour la recouvrir totalement, nous avons donc créé une page à la taille exacte de la photo de la personne. Les images de paires de lunettes peuvent alors être placées comme un graphique normal et être superposées à l'image de fond, après avoir rendu transparente la couleur entourant la paire. Notons enfin qu'il est techniquement impossible de rendre les verres semi-transparents. Nous entendons par là qu'il aurait été plus agréable de voir les yeux du visage au travers des lunettes mais qu'il est n'est pas possible, du moins en HTML, d'obtenir cette fonctionnalité.

Afin de faciliter la programmation d'applications de type commerce électronique, le langage HeiTML fournit la possibilité de créer une « session ». Grâce à cette option, il est possible de garder en mémoire une série de variables globales dont les valeurs pourront être utilisées de page en page sans pour autant devoir les poster (c.-à-d. les envoyer à la page suivante via le réseau). Au début de la session, un identifiant est créé, il suffit au programmeur de poster cette information à la page HeiTML suivante dans la session, pour ensuite pouvoir accéder aux variables globales. Une session est donc créée à l'inscription du client et les différentes informations sont placées dans des variables globales.

[24] Notons que la société Netscape est en train de spécifier des extensions au langage HTML qui permettront pour les prochaines versions de son navigateur le travail en couches et donc la superposition d'images ou de textes.

4.5.4.3 *Phase de tests*

La phase de tests est classique et consiste à vérifier si l'application fon répond à toutes les exigences requises. Notre test, devant recouvrir toutes le l'application, est repris comme exemple indicatif et est commenté dan suivant.

4.6 Descriptif de l'application

Nous proposons dans cette section différentes captures d'écran tirées de l'exécution de notre application. Un descriptif explique les différentes étapes et choix effectués pour l'exemple.

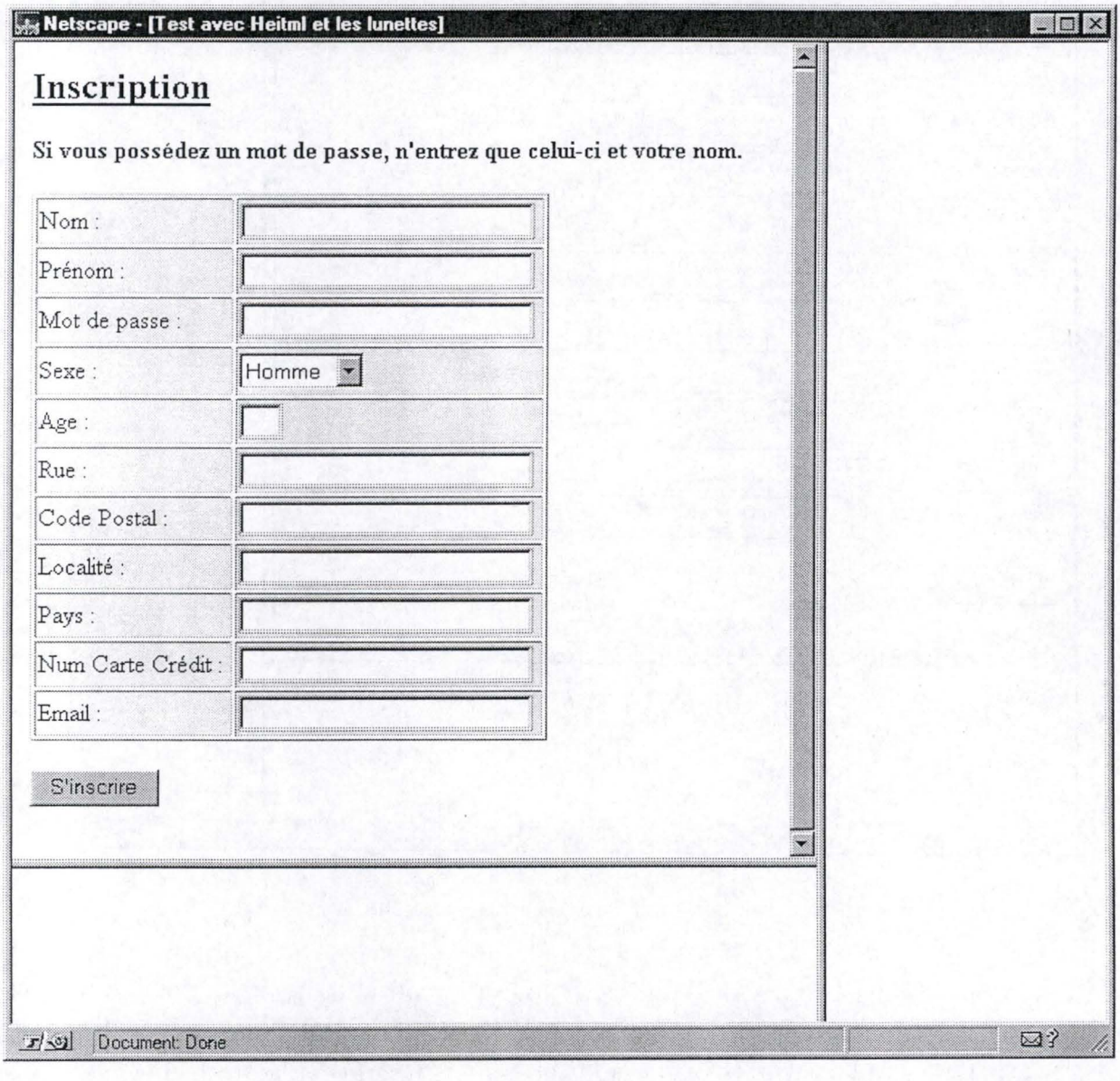


Figure 4-10 : Page d'inscription

En entrant l'URL de la première page de notre application, l'utilisateur est invité à s'inscrire comme client potentiel (Figure 4-10). Il doit pour cela introduire les différentes données le concernant. Il est donc invité à donner les informations à son propos, nécessaires à l'éventuel acheminement de la paire de lunettes qu'il pourrait acheter.

Notons que si le client est une personne qui a déjà utilisé l'application, il a reçu un mot de passe qui lui permet de passer outre cette phase d'identification. Il peut alors introduire uniquement son nom et son mot de passe. Mais nous reviendrons sur cette possibilité.

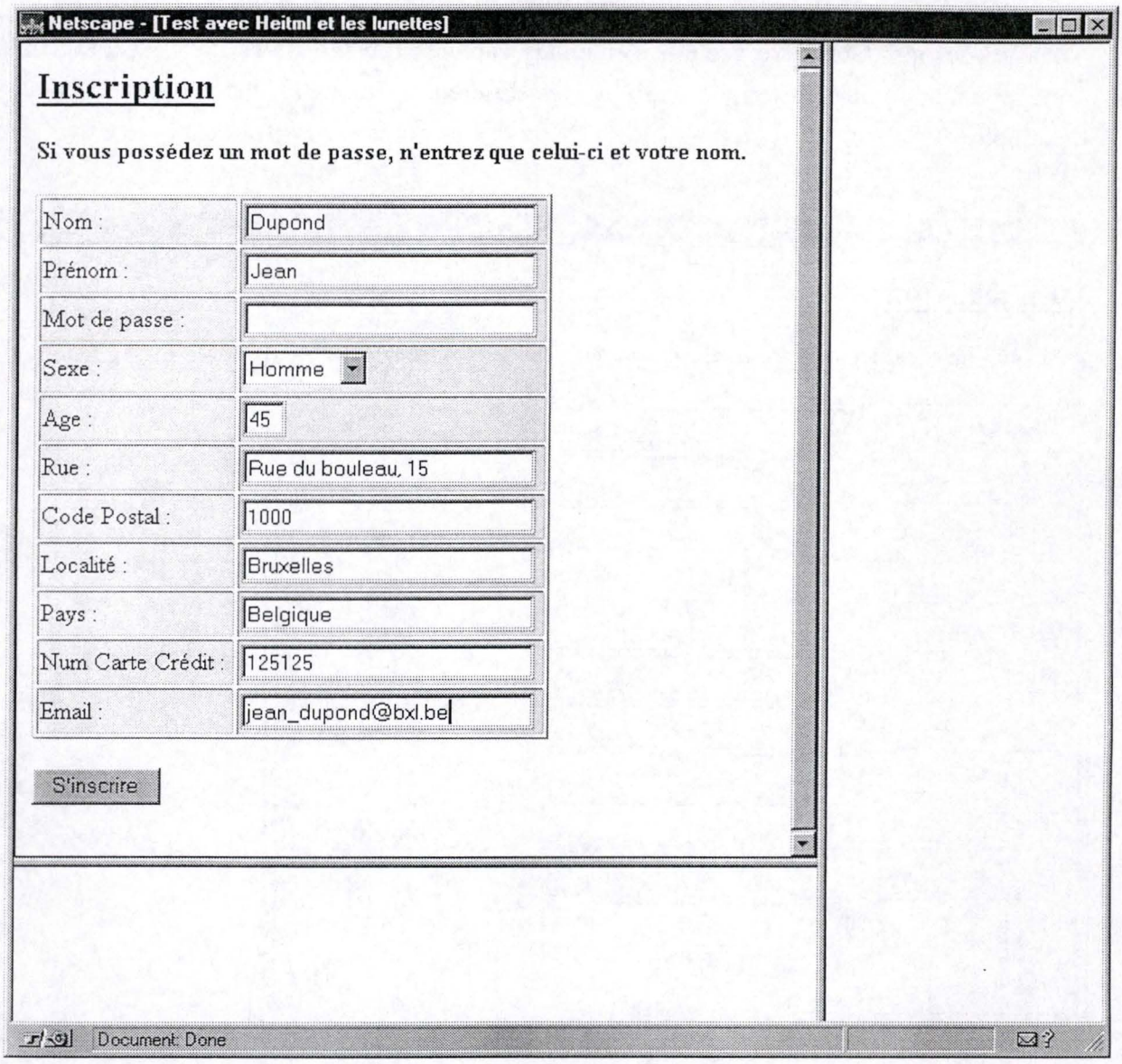


Figure 4-11 : Inscription d'un nouvel utilisateur

Nous trouvons dans le cas où l'utilisateur utilise l'application pour la première fois et remplit donc toute la fiche (sauf le mot de passe qu'il ne possède pas encore). Une fois celle-ci remplie, il lui suffit de cliquer sur le bouton « S'inscrire » (Figure 4-11).

La page suivante utilise les données afin de vérifier si le client existe déjà ou non. L'algorithme complet pour vérifier cela est le suivant :

```
Si client existe (c.-à-d. Nom et Password déjà dans la base)
Alors Afficher les précédents achats éventuels et proposer une nouvelle sélection
Sinon Vérifier les données (tous les champs remplis et adresse électronique valide)
        Si Données correctes
        Alors Enregistrer l'utilisateur dans la base et proposer la sélection
        Sinon Signaler l'erreur et proposer au client de recommencer son inscription
```

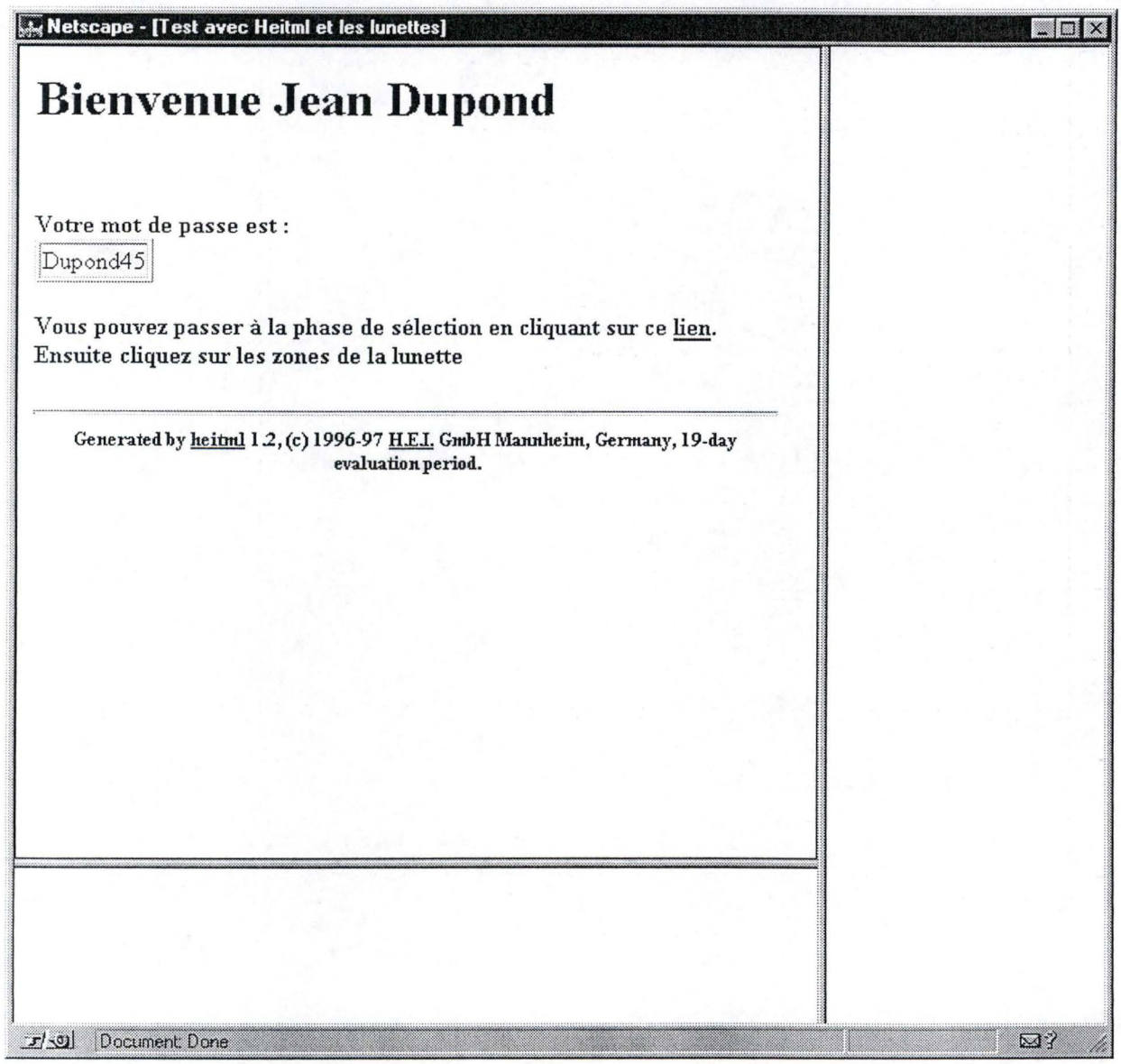


Figure 4-12 : Accueil du nouvel utilisateur

Nous nous trouvons donc dans le second cas de l'algorithme. Ce nouvel utilisateur est alors accueilli par cette page (Figure 4-12) qui lui propose son mot de passe et l'invite à passer à la phase de sélection d'une paire de lunettes solaires. Le client est à ce moment enregistré dans la base de données.

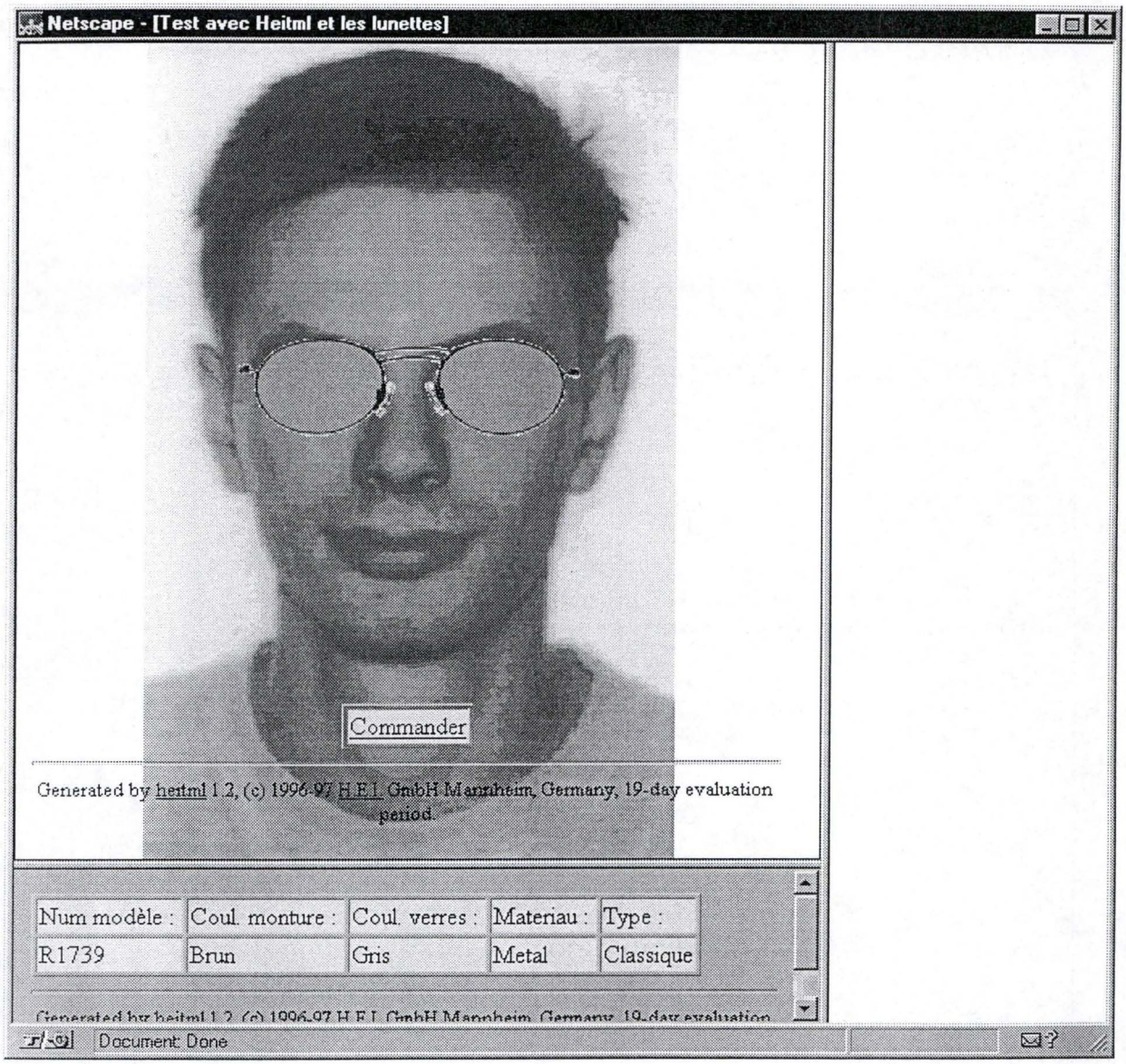


Figure 4-13 : Première page pour la sélection d'une paire de lunettes solaires

Basée sur les critères de sexe et d'âge du client, une première paire de lunettes est proposée via cette page (Figure 4-13). En effet, une monture est soit destinée à un homme, soit à une femme ou est unisexe. De même, une monture moderne est plutôt destinée à une tranche d'âge inférieure à 30 ans, une monture classique à des clients ayant entre 30 et 50 ans, les montures traditionnelles étant plutôt destinées à la tranche d'âge supérieure à 50 ans. Les caractéristiques plus complètes de cette première paire sont affichées au bas de la page. Le client peut alors cliquer sur les zones définies sur la paire de lunettes, pour en changer les caractéristiques. Il peut à ce moment lire à quelle caractéristique est liée la zone, grâce au texte affiché dans la barre de statut[25] de la fenêtre. Cliquons par exemple sur un des verres de la paire pour en changer la couleur.

Figure 4-14 : Proposition de couleurs de verres

[25] partie inférieure de la fenêtre physique de l'application dans laquelle peut être affichée du texte.

A ce moment, une série de propositions de paires de lunettes est affichée dans la partie droite de la fenêtre (Figure 4-14). Ces différentes paires ont des caractéristiques identiques à celle affichée dans la fenêtre principale, à l'exception de la couleur des verres. L'utilisateur peut alors soit choisir une autre caractéristique en cliquant sur une autre zone de la lunette, soit choisir une paire parmi les propositions. Prenons ce dernier cas en cliquant sur la paire ayant les verres de couleur verte.

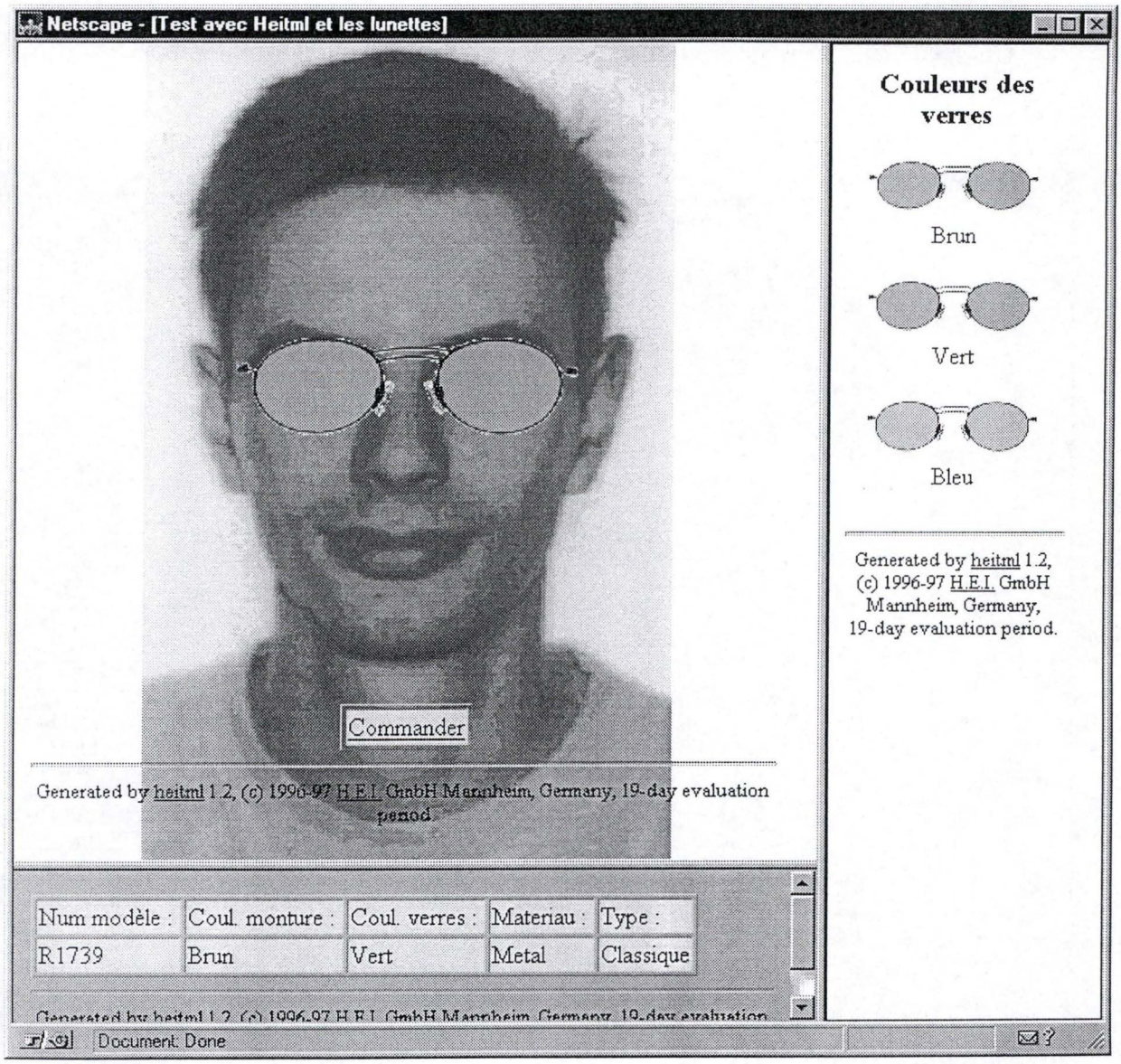


Figure 4-15 : Choix d'une nouvelle couleur de verres

La paire choisie prend alors la place de la précédente dans la fenêtre principale et les caractéristiques de la nouvelle sélection s'affichent en bas de l'écran (Figure 4-15). Le cycle recommence jusqu'à ce que le client appuie sur « Commander ». Mais cette fois, prenons l'exemple où l'utilisateur clique l'entre-nez[26] afin de choisir le modèle de la monture.

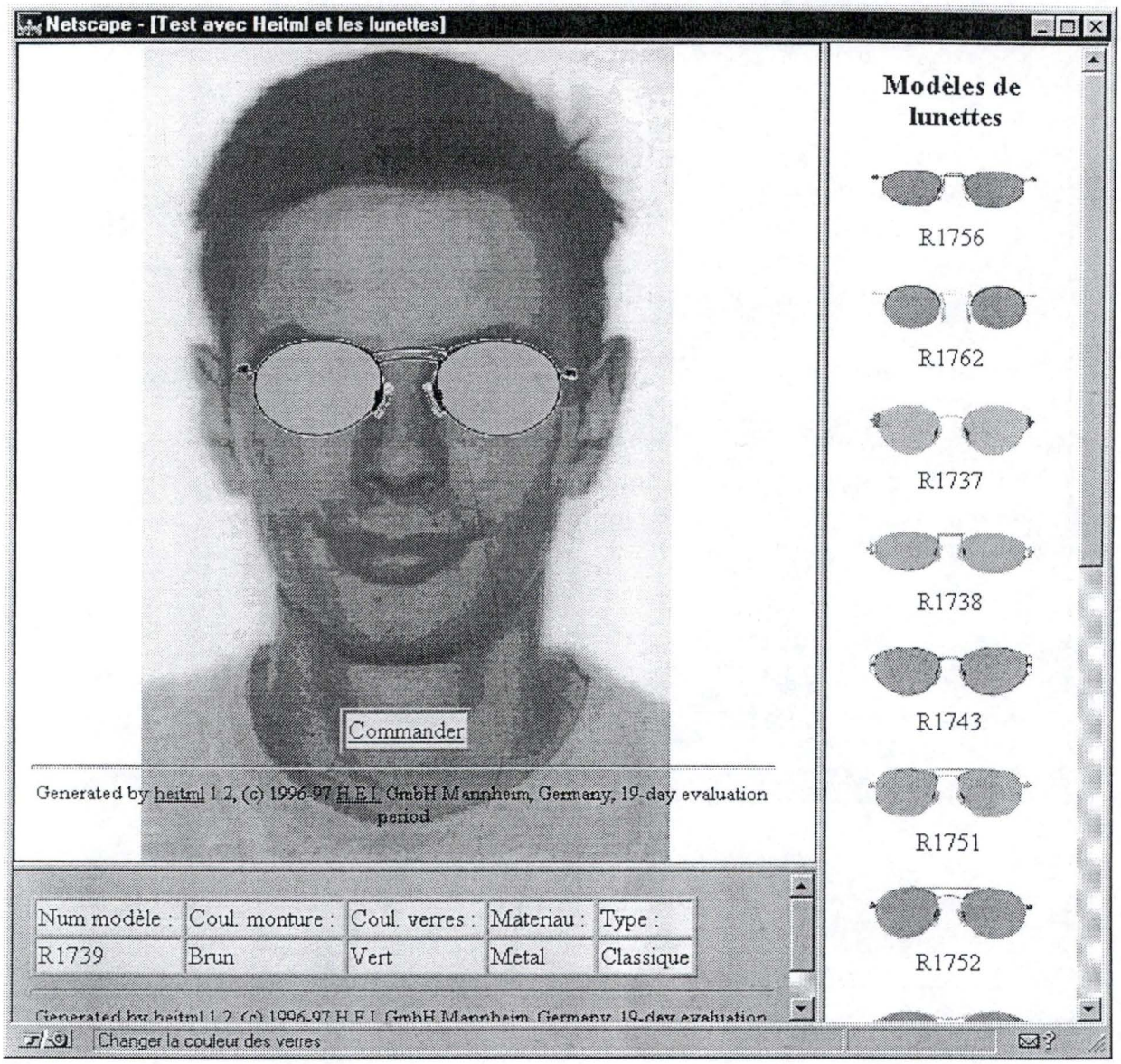


Figure 4-16 : Proposition de modèles de lunettes

Une liste de modèles différents ayant la même couleur de verres est affichée et l'utilisateur clique sur un autre modèle que celui affiché (Figure 4-16). Mais le client préfère le premier modèle et décide alors de commander cette paire en cliquant sur le lien créé dans ce but.

[26] L'entre-nez est un terme technique utilisé en optique donné à la partie de la monture reliant les deux verres.

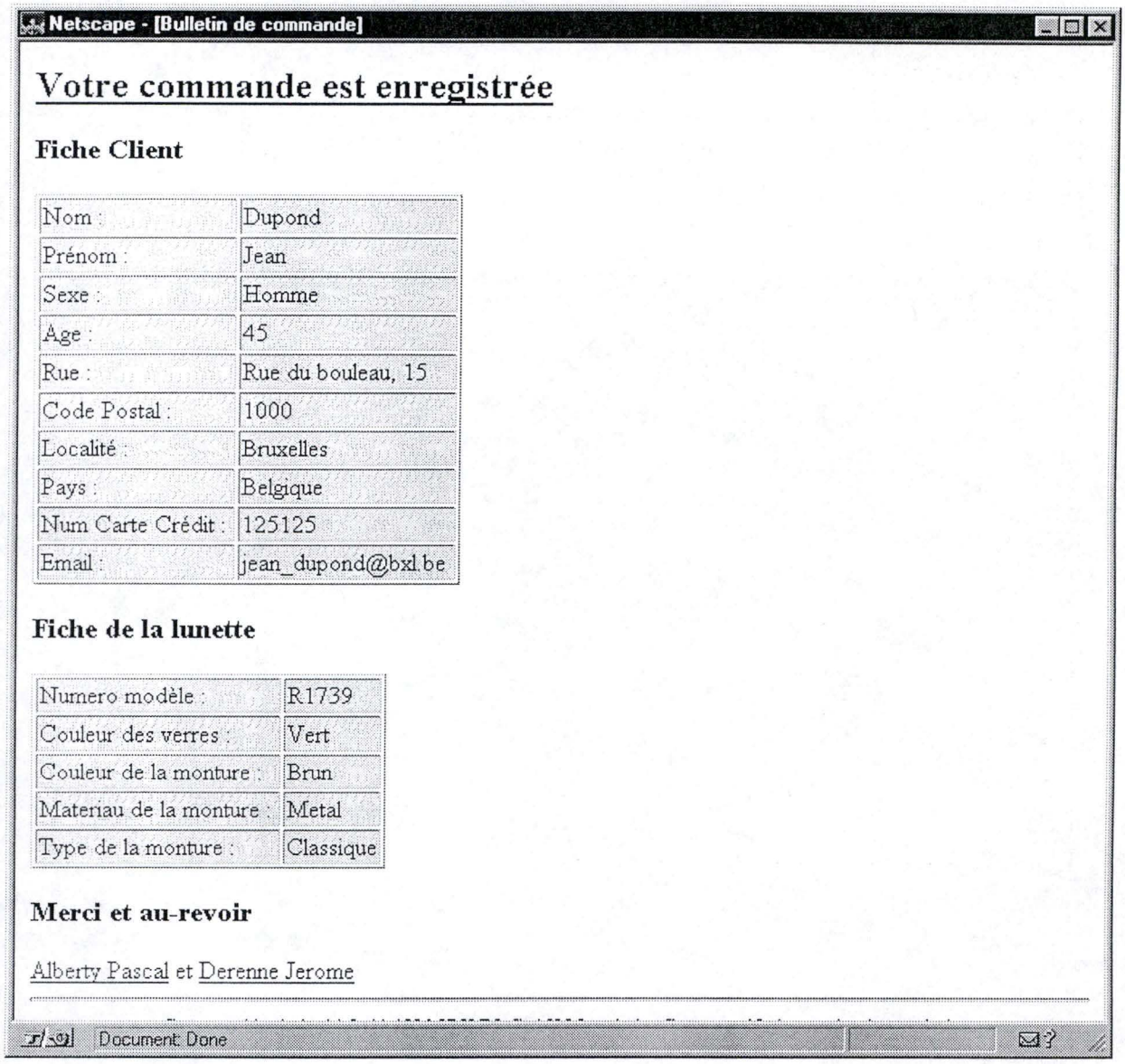


Figure 4-17 : Confirmation de l'enregistrement d'une commande

A ce moment, les caractéristiques concernant l'achat du client sont enregistrées dans la base de données et confirmées au client via cette page (Figure 4-17) reprenant les informations le concernant ainsi que les caractéristiques de la paire de lunettes solaires choisie.

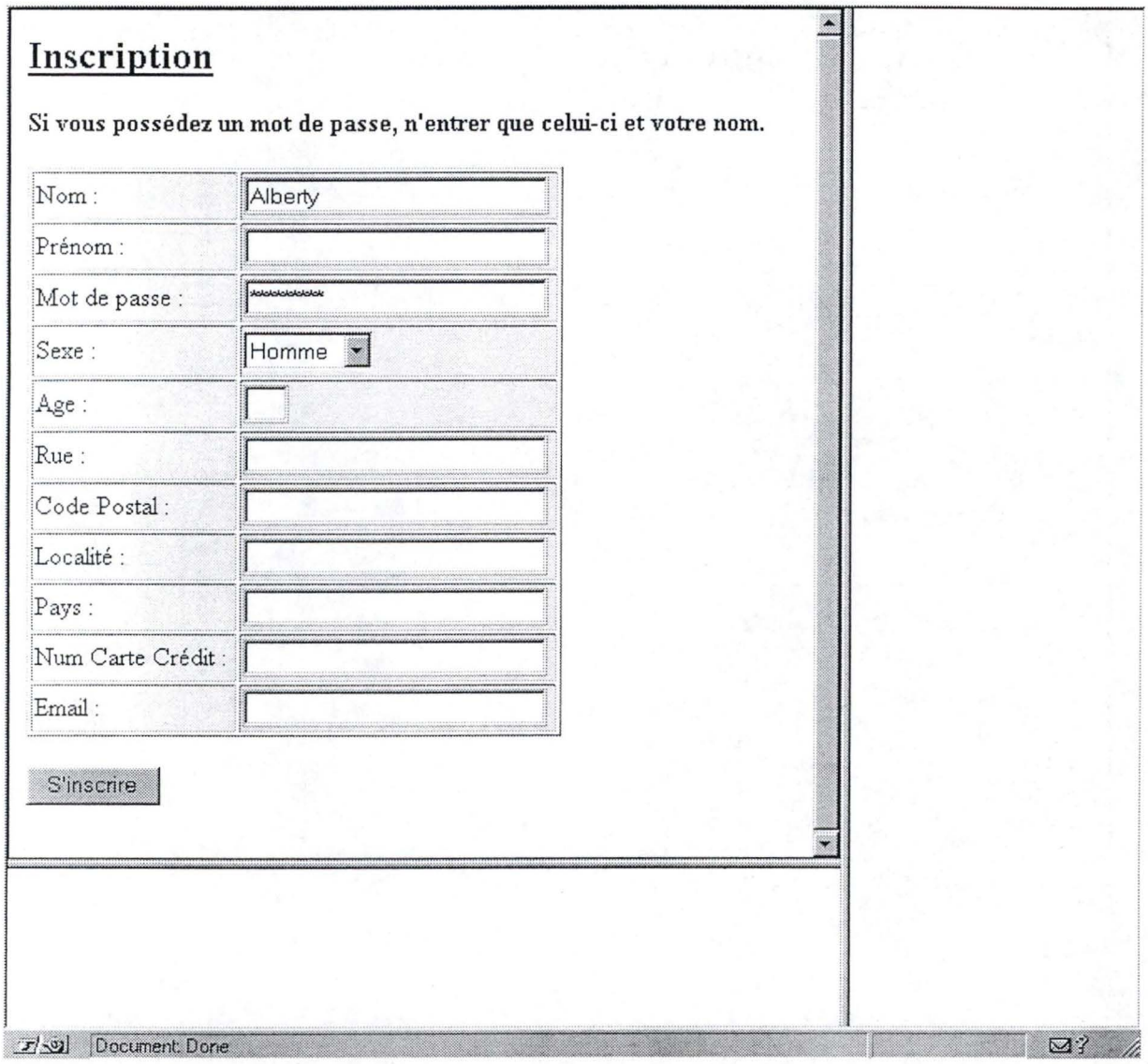


Figure 4-18 : Inscription d'un ancien client

Revenons au le cas où le client est une personne ayant déjà utilisé le programme. Il possède alors déjà un mot de passe. Il peut donc passer outre cette phase en introduisant uniquement celui-ci et son nom de famille (Figure 4-18).

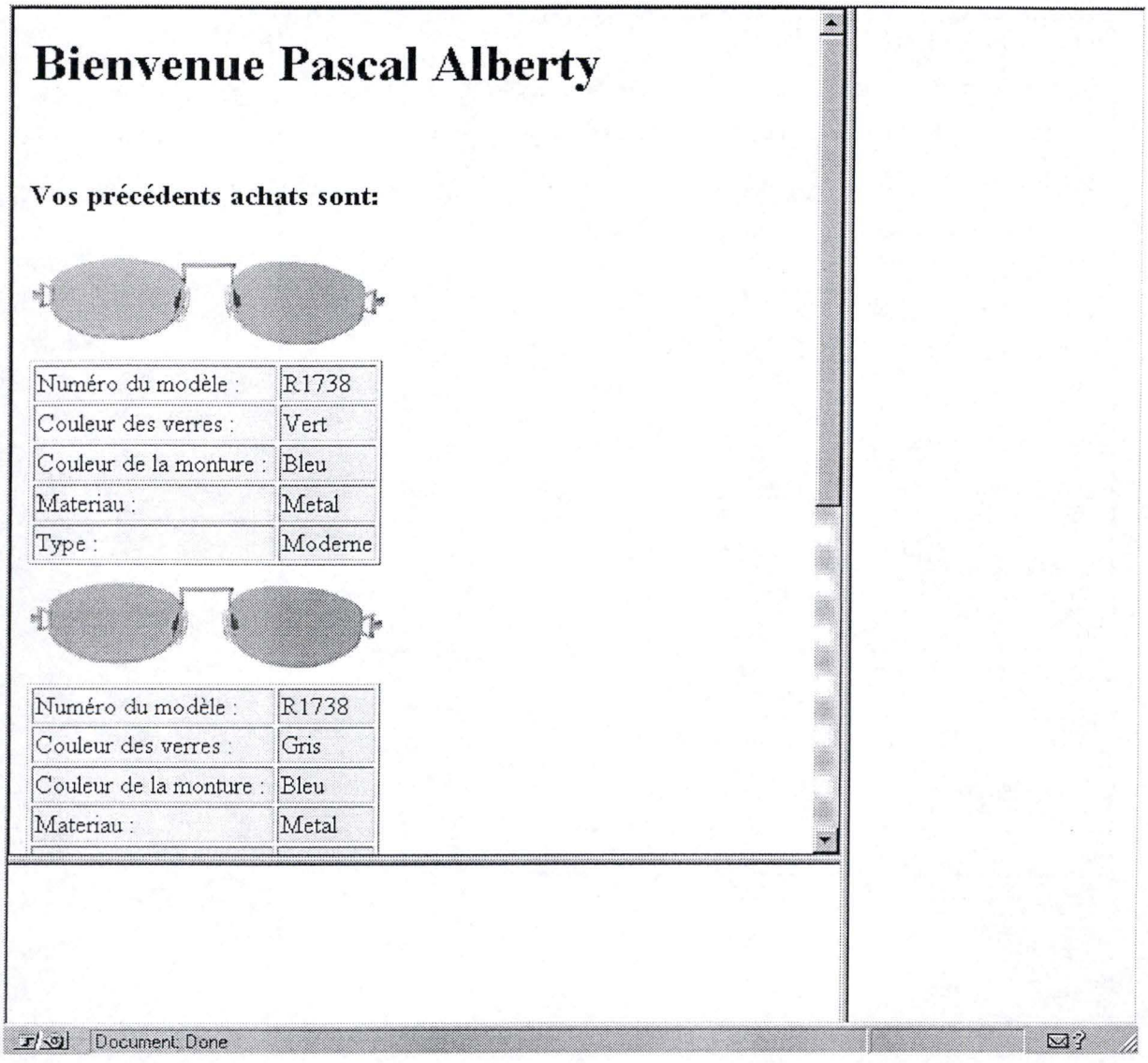


Figure 4-19 : Rappel des précédents achats d'un ancien client

L'utilisateur peut alors visionner ses anciens achats avant d'être de nouveau invité à passer à la phase de sélection, identique à celle précédemment expliquée (Figure 4-19).

4.7 Critiques et propositions d'amélioration

Critiquons la construction de l'application en reprenant ses différentes phases, à savoir la modélisation du problème, les options de l'architecture et la technique de recherche, ainsi que les détails d'implémentation.

4.7.1 Modélisation

Nous n'avons pas réellement employé l'extension multimédia du schéma ERA proposée. En effet, l'attribut « photo » de la lunette et du client n'a pas été traité comme tel. Cependant, nous nous sommes permis d'en utiliser une partie en stockant uniquement le nom du fichier car, dans notre cas, la ressource multimédia n'est que d'un seul type, à savoir une image graphique. Sa traduction en termes d'implémentation HTML est donc bien définie. Dans le cas où l'attribut aurait pris la « valeur » d'un son ou d'une vidéo, le complément d'information donnant son type aurait été utile.

4.7.2 Architecture et technique de recherche

Il ne nous a pas été possible d'utiliser les modèles des méthodologies étudiées précédemment. Si les primitives nous permettent de modéliser la navigation entre les pages simples, elles ne nous permettent pas de modéliser en particulier l'étape de sélection de la paire de lunettes. Celle-ci aurait pu être en partie modélisée par la notion d'index conditionnel de RMM en proposant une liste de valeurs tirées de la base de données et correspondant à une certaine condition (la liste de lunettes ayant la même couleur de verres). Mais, comme déjà signalé, il est impossible de modéliser une recherche sur plusieurs arguments. De même, nous ne savons pas modéliser la technique de recherche à savoir le fait que le choix d'une des informations présentées doit influencer les informations de la page de retour de l'index.

Nous pouvons aussi critiquer notre choix proposant de relier chaque caractéristique de la paire de lunettes à un élément de l'objet graphique. On pourrait imaginer un système combinant le choix via les zones sensibles et des éléments textes traditionnels. L'algorithme de recherche pourrait être remplacé par un système de construction de requêtes dynamiques.

4.7.3 Implémentation

Notre exemple d'application a été choisi simple. Nous pourrions lui apporter les améliorations suivantes. L'introduction de la photo du client qui servira d'image de fond pour la page de sélection. Cette option a été envisagée mais aurait rendu la partie technique de l'implémentation beaucoup plus compliquée, et également complexifié la procédure d'inscription du client qui est supposé ne pas être obligatoirement spécialiste en informatique. Il faudrait effectivement que la photo soit parfaitement formatée, détail technique important.

Nous nous sommes également limités à deux caractéristiques pour le choix d'une paire de lunettes : la couleur des verres et le modèle des montures. Il est évident que cette option peut être étendue à d'autres données comme le matériau utilisé, la couleur de la monture, la taille des branches. Dans ce but, l'ajout d'une page représentant le client et la paire de lunettes vus de profil serait utile. En effet, n'oublions pas que chaque caractéristique doit avoir son équivalent en zone sensible sur l'image de l'objet à acheter.

4.8 Conclusion

Ce chapitre s'inspire directement du chapitre trois afin d'en tirer une application à part entière, reposant sur les diverses analyses présentées. Comme nous l'avons déjà mentionné, cet exemple met en évidence l'importance de l'existence de liens entre la représentation interne et la représentation externe du problème.

L'interface et la modélisation repose entièrement sur le choix de l'application de gestion choisie et appliquer une méthodologie tel que RMM ou SHDT s'avère en pratique partiellement impossible, étant donné le fort lien d'interaction entre les éléments présentés à l'utilisateur. Dans cet exemple, la modélisation de notre problème s'est effectuée pas à pas, aucune méthodologie étudiée n'a été suivie, et comme on peut le voir, la gestion et la solution du problème reposent en partie sur l'aspect technique et sont directement liées au domaine d'application.

Nous sommes alors en droit de nous demander si malgré ces différents points, il n'existe pas un aspect méthodologique aidant à la résolution de tels problèmes. RMM et SHDT s'avèrent fort inadéquats mais ne pourrait-on pas en tirer les enseignements, modéliser cette classe de problèmes et trouver les concepts et les règles relatives pour ces modèles ? Nous essayons dans le chapitre 5 de répondre à ces questions. Nous esquissons les points principaux d'une telle méthodologie et justifions l'emploi de nouvelles règles et suivant les cas, la non-utilisation des anciennes règles présentes dans d'autres méthodologies.

Chapitre 5
Conclusion

A la suite de notre expérience pratique, on est en droit de se demander quels sont les ingrédients réutilisables, intéressants et utiles pour une proposition d'une nouvelle méthodologie complète de conception d'applications Web, formés de formulaires multimédias et basées sur un schéma conceptuel. En conséquence, nous proposons dans ce dernier chapitre des éléments de réflexion constituant la base d'une méthodologie qui permettrait de résoudre notre classe de problèmes étendus. Nous les décrivons par rapport aux trois pôles constituant une méthodologie (modèles, démarche et outils) et expliquons pourquoi ils ont été choisis. Enfin, nous en justifions le contenu et les situons par rapport aux méthodologies et outils décrits dans les chapitres précédents.

5.1 Eléments de réflexion pour une méthodologie de conception d'applications Web composées de formulaires multimédias

Comme spécifié plus tôt, si l'on désire définir une méthodologie complète, il est nécessaire d'en définir les composants suivant trois pôles. Le premier concerne les modèles de données ; ceux-ci permettent, selon leur nature, de capter différentes notions par une modélisation précise et complète devant couvrir tous les aspects de la future application. Nous proposons trois types de modèles à savoir le schéma conceptuel, le modèle navigationnel et les contextes navigationnels. Le second pôle traite de la démarche proposée par la méthodologie. Elle décrit les règles à suivre lors de la modélisation de l'application. Enfin, le troisième parle d'outils supportant la méthodologie et devant alors répondre à certains critères que nous précisons. Les trois sous-sections suivantes citent, pour ces trois pôles, les types généraux des éléments qui les composent. Comme le lecteur peut le remarquer, nous nous limitons à décrire les phases de design. Il est évident que ces phases sont précédées d'une analyse des besoins, et suivies par l'implémentation de l'application.

5.1.1 Pôle modèle

Le schéma conceptuel. L'utilisation d'un schéma conceptuel de base de données permet d'obtenir une modélisation de la représentation interne du problème, afin de capturer la sémantique du domaine d'une façon aussi abstraite que possible.

Le modèle navigationnel. Un modèle navigationnel permet de modéliser au mieux l'interaction offerte à l'utilisateur. Il permet de représenter la manière dont les informations sont structurées et présentées à l'écran. Il se compose de primitives permettant de modéliser cette structuration et navigation. L'utilisateur final, dès lors qu'il utilisera l'application, sera en mesure de naviguer parmi les diverses informations disponibles suivant l'analyse effectuée par le concepteur.

Les contextes navigationnels. La prise en compte de contextes navigationnels est nécessaire pour permettre de structurer et de modéliser plus précisément les possibilités d'accès à l'information selon le contexte d'accès choisi par l'utilisateur, au contraire du schéma navigationnel qui ne représente que la navigation envisagée pour l'accès aux différents groupes d'informations de l'hypermédia. L'utilisation de ces contextes navigationnels permet une plus grande structuration des informations selon la méthode de recherche choisie par le concepteur de l'application.

5.1.2 Pôle démarche

La démarche méthodologique que nous considérons est basée sur trois étapes de design.

Le design conceptuel. La première étape concerne le design conceptuel ; il permettant de modéliser les données utilisées par l'application en se utilisant le schéma conceptuel. Cette phase répond à des règles fixes et définies selon le modèle choisi.

Le design navigationnel. Dans cette deuxième étape, la navigation au sein des informations est modélisée. Celle-ci est en effet nécessaire afin d'obtenir une interface correspondant au mieux aux désirs de l'utilisateur, tant du point de vue de l'interaction que de celui de la navigation. Elle peut se baser sur le schéma conceptuel modifié à l'aide de règles et de primitives représentant la navigation.

Le design contextuel. Enfin, cette troisième étape modélise les particularités de la navigation en prenant compte, par exemple, des éléments d'interaction et des techniques de recherche utilisés et nécessaires au type d'application choisi.

Bien que nous nous limitions à ces phases, il est utile de préciser que leur résultat doit être intégré selon des règles à définir, et de cette intégration, doit être produite, également via des règles de transformations, l'application prévue pour une plate-forme donnée.

5.1.3 Pôle outil

Les différentes phases de la méthodologie doivent être supportées par un ou plusieurs outils permettant la dérivation finale de l'application. Il est nécessaire de disposer d'outils prenant en charge le design des schémas conceptuel, navigationnel et contextuel. Les différentes règles de transformation permettant le passage d'une phase à l'autre devront être également intégrées.

5.2 Description des éléments choisis

Les types des éléments de l'éventuelle méthodologie étant cités, il reste maintenant à les décrire en donnant des pistes quant à leur contenu, leur utilité et, si possible, leur origine. Les sous-sections suivantes se proposent de réaliser cela, en reprenant ces éléments avec la même classification basée sur les trois pôles.

5.2.1 Pôle modèle

Le schéma conceptuel. Le modèle du schéma conceptuel ERA habituel est proposé [BODART93a]. Cependant, les éléments multimédias doivent être pris en compte : on y insérera par exemple, les attributs de type Mime proposé précédemment (cfr. sous-section 3.2.1).

Comme nous l'avons déjà mentionné aux chapitres précédents, le design d'applications hypermédias doit considérer différents points de vue tels que la structuration de l'information, le design navigationnel et le design de l'interface. Le design d'hypermédias classique est très proche du design de bases de données, car tous deux travaillent avec des informations hautement structurées et volatiles, comme cela est souvent le cas dans le design d'applications pour le commerce électronique. Le modèle ERA proposé présente l'avantage d'être facilement utilisable, bien décrit et, de par son utilisation répandue, il est facilement réalisable à l'aide des règles bien définies.

Actuellement, de plus en plus d'outils supportent la phase de conception de bases de donnée par l'intermédiaire du design de schéma ERA ou relationnel. Celui-ci est facilement dérivable manuellement ou à l'aide d'outils existants (par exemple DBMain). En effet, les étapes permettant cette dérivation sont bien spécifiées. Le passage par cette étape de design ERA est très importante : il permet au concepteur de l'application de s'abstraire des détails techniques de l'implémentation liés à la résolution de son problème.

Parmi les méthodologies que nous avons étudiées, peu se basent sur un schéma conceptuel de type ERA. RMM utilise un schéma relationnel proche de celui-ci. D'autre part, la majeure partie des autres méthodologies se basent sur des schémas orientés objet ou propriétaires, certains d'entre eux étant plus ou moins transposables en un schéma ERA.

Le modèle navigationnel. Le modèle navigationnel proposé doit permettre de modéliser l'architecture globale de l'application ainsi que les caractéristiques particulières au monde du commerce électronique. Afin d'être efficace dans un maximum de cas, cette phase utilise un modèle navigationnel incluant la notion de page, de menu, d'index, de formulaire, avec pour chacun de ces concepts un équivalent « dynamique » (c'est-à-dire une page générée automatiquement mais pouvant contenir différentes informations à chaque visualisation) ; les concepts de tour guidé et de tour guidé indexé doivent également faire partie du modèle.

Des liens hypermédias renvoyant vers d'autres menus, d'autres index ou d'autres éléments multimédias doivent exister dans ce modèle. Les liens uni- et bidirectionnels sont proposés.

Notre choix d'un modèle navigationnel repose donc en partie sur ceux présentés dans RMM et SHDT. Afin de modéliser l'architecture globale de l'application (c'est-à-dire la navigation simple entre pages statiques), les primitives de SHDT peuvent être utilisées. De même pour la modélisation de la navigation au sein d'informations plus structurées et se basant sur un schéma conceptuel, celles de RMM sont alors prise en compte. En effet, la combinaison de ces primitives permet d'élargir l'utilité du modèle proposé par RMM à la modélisation d'un site plus complet, et d'élargir celui de SHDT à des informations plus structurées.

Il est donc nécessaire d'intégrer ces deux modèles en un modèle plus large reprenant les concepts cités. Ceci permettrait d'avoir un niveau plus élevé de structuration et de modélisation des accès aux informations, en ce qui concerne les étapes de l'application présentant une interaction et une navigation plus complexes. En outre, il est nécessaire de posséder un modèle nous permettant de modéliser des caractéristiques plus particulières au type d'applications liées au commerce électronique.

Les contextes navigationnels. Les contextes navigationnels sont utiles afin que le concepteur de l'application adopte une vue de niveau plus avancé, et possède une modélisation intégrant davantage les techniques de recherche choisies dans le cadre d'applications de type commerce électronique.

En effet, comme nous avons pu le constater via notre étude de cas, il n'est pas possible de modéliser simplement la structure navigationnelle d'une étape comme le choix d'une paire de lunettes. Celle-ci dépend de la composition de l'objet, de la plate-forme utilisée et de la technique utilisée, dont dépendront à leur tour les techniques d'interaction implémentées par l'application finale.

Nous proposons, à l'aide de la notion de contextes navigationnels, de capter ces aspects dans un nouveau modèle. Dès lors, les détails d'implémentation des techniques employées sont modélisés via les primitives des contextes navigationnels en un niveau plus abstrait. Par exemple, il sera possible de cacher les notions telles que les images cliquables utilisées dans notre étude de cas. De même, toute la partie recherche pourrait être modélisée à l'aide de contextes navigationnels puisque ceux-ci prennent en compte la technique d'accès aux informations disponibles, leurs interactions avec ces mêmes informations et leur affichage à l'écran. Il est certain que la création ou non de contextes navigationnels dépend également des technologies employées (ils seront structurés différemment selon les types multimédias rencontrés).

En matière de commerce électronique, une très importante part de la navigation que réalisera l'utilisateur sera destinée à effectuer une recherche sur ce qu'il désire acheter. C'est pourquoi la technique de recherche prise en compte n'est pas à négliger. Par exemple, le concept de Queries Dynamiques peut être une bonne option, tout comme l'approche que nous avons adoptée lors de notre étude de cas sur la vente de lunettes de soleil. De même, les contextes navigationnels modélisent l'accès aux différents types de lunettes et aux couleurs de lunettes suivant que l'utilisateur clique sur telle ou telle partie de l'objet. Il change donc de contexte navigationnel de par sa navigation. Le design navigationnel et la modélisation de différents contextes doivent prendre en compte ce choix concernant la manière dont la recherche est effectuée, car ce sera toute l'interaction et l'ergonomie de l'application qui seront influencées par celle-ci.

5.2.2 Pôle démarche

Le design conceptuel. La procédure de design ERA consiste à représenter le domaine d'information de l'application à travers un schéma ERA. La procédure utilisée lors de la phase de design du schéma conceptuel de l'application est celle employée habituellement, excepté que l'on tiendra compte des caractéristiques propres à l'utilisation de moyens multimédias (en y insérant le type Mime par exemple).

Le design navigationnel. La phase de design navigationnel, quant à elle, consiste non seulement à modéliser les chemins qui rendront la navigation possible mais aussi à établir une interface convenant au mieux à l'attente des utilisateurs. Les règles de transformation appliquées au schéma ERA obtenu à l'étape précédente doivent aboutir au schéma navigationnel qui est alors augmenté des primitives navigationnelles. Les étapes de cette phase reposent sur celles des méthodologies RMM et SHDT. Cependant, et comme pour les modèles, une certaine intégration est nécessaire afin de prendre en compte les différences de ces dernières.

Le design contextuel. La modélisation des contextes navigationnels doit prendre en compte les différents aspects définis par le modèle et ses primitives. La description de cette phase peut faire l'objet d'une étude ultérieure.

Le schéma suivant présente l'enchaînement des différentes phases définies ci-dessus :

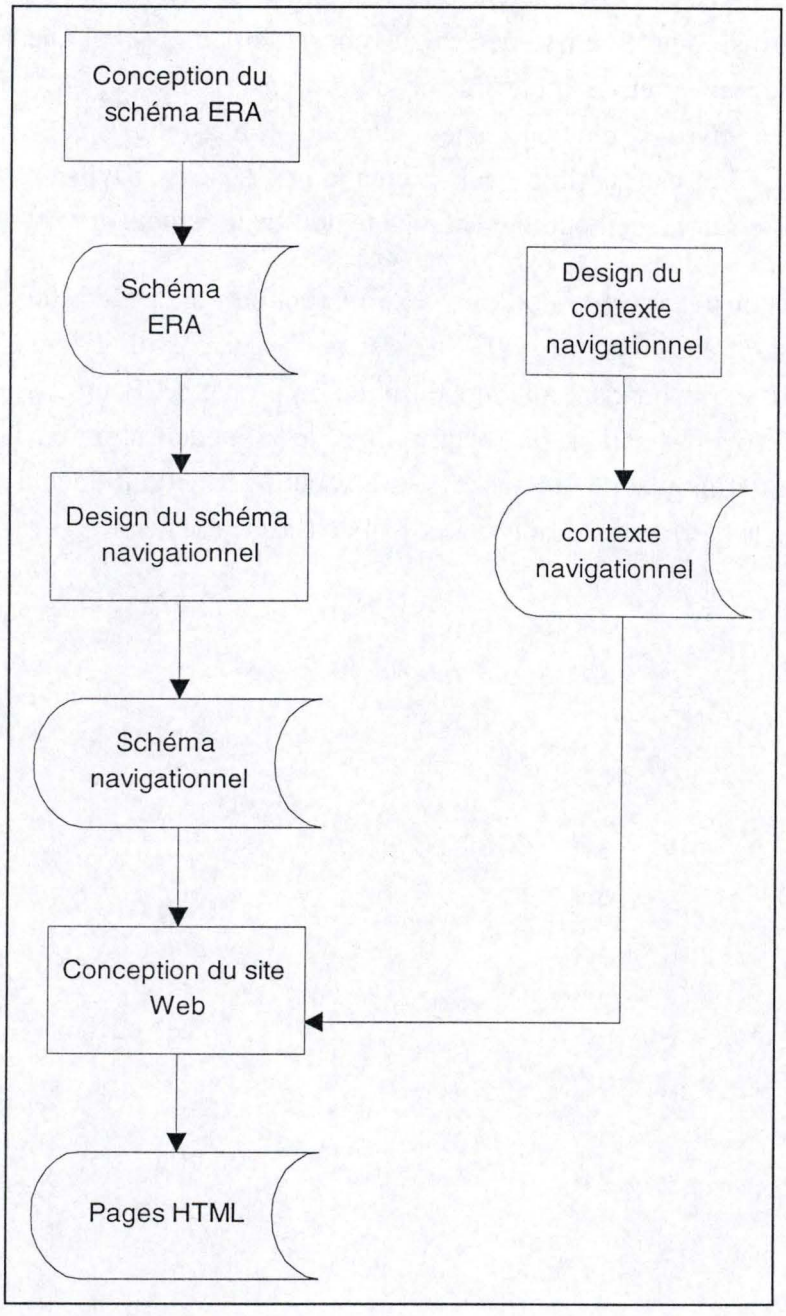


Figure 5-1 : Proposition d'une méthodologie

5.2.3 Pôle outil

Pour certaines étapes, il existe divers outils qui peuvent servir de support à la future méthodologie. Ainsi, pour la phase de design conceptuel, un outil tel que DBMain est très utile. Il se charge en effet de toute l'étape de design de schéma ERA. D'autres outils, implémentés pour diverses méthodologies (tels que RMCase ou WebDesigner) montrent qu'une automatisation est possible pour la création de schéma navigationnel. Cependant, l'outil qui supporterait la méthodologie complète doit tenir compte des modèles précités.

Afin de ne pas nous écarter de notre contexte particulier, à savoir celui des applications hypermédias dans l'environnement HTML, cet outil doit prévoir la génération complète des pages HTML correspondant à la modélisation du problème. Pour cela, un ensemble de règles de transformation partant du schéma final de la modélisation en différentes pages HTML doit être défini et supporté par cet outil. L'outil permettrait donc de créer les pages HTML ainsi que les différents scripts d'accès aux bases de données.

Conclusion

Dans un premier chapitre, nous avons présenté un état de l'art de méthodologies de conception d'applications hypermédias suivant leurs trois pôles (modèles, démarche et outils), des outils de génération ou d'édition de pages HTML, ainsi que des techniques d'interaction susceptibles d'être rencontrées dans ce type d'applications.

Ensuite, nous avons étudié une classe de problèmes simples, la création de formulaires simples pour le Web, à partir d'un schéma conceptuel. Nous avons alors remarqué que les méthodologies et outils existants permettaient de produire une solution relativement acceptable ; nous avons également présenté des avantages et lacunes pour les différentes méthodologies étudiées.

En étendant certaines caractéristiques de cette première classe de problèmes, nous avons proposé, dans un troisième chapitre, une classe plus complexe. Ces extensions portaient essentiellement sur les caractères multimédia et graphique de l'interface et autorisaient plus de convivialité et une interaction plus poussée.

Afin d'illustrer et d'analyser cette classe de problèmes étendue, nous avons consacré le chapitre suivant à un exemple concret de construction d'une application graphique multimédia pour le Web.

Les différentes analyses effectuées nous ont permis de mettre en évidence, pour chaque pôle de la méthodologie, les éléments à considérer afin de développer une éventuelle méthodologie de conception d'applications appartenant à la classe de problèmes étendue. En ce qui concerne les modèles, nous proposons l'utilisation du schéma ERA augmenté des attributs de type Mime, de l'intégration des modèles navigationnels de RMM et SHDT, ainsi que des contextes navigationnels, ces derniers restant à définir. Les phases de la démarche à effectuer seraient constituées de la modélisation conceptuelle du schéma ERA, d'une étape intégrant les phases de design navigationnel de RMM et SHDT, et de règles à déterminer pour la modélisation des contextes navigationnels. Enfin, les outils devraient supporter les modèles et la démarche, et permettre la génération d'applications interactives développées pour la plate-forme Internet, dans le domaine particulier du commerce électronique.

Ces considérations nous permettent de supposer qu'il serait possible, d'après les pistes proposées, de développer une méthodologie adaptée.

Tables des figures et tableaux

Bibliographie

[BARNES96] Helen BARNES et Mike DWYER, *Designer/2000 Web Enabling Your Applications*, documentation commerciale, Oracle, mars 96.

[BICHLER96] M. BICHLER et S. NUSSER, *Developping Structured WWW-Sites with SHDT*, http://wwwi.wu-wien.ac.at/shdt/wwwpaper/shdt.html, date de visite dernière du lien : octobre 96.

[BODART93a] F. BODART et Y. PIGNEUR, *Conception assistée de systèmes d'information*, 2ème édition, Edition Masson, 1993.

[BODART93b] F. BODART et J. VANDERDONCKT, *Conception d'Interfaces Homme-Machine*, cours universitaire de 1ère licence en informatique, Institut d'Informatique, FUNDP, Namur, 1994.

[EKTOS97] *Ektos Information Managment - Base de données Multimédia orientée objet pour Ms-Windows et le Web*, documentation commerciale, Ektelis, Belgique, 1997.

[EVRARD96] C. EVRARD et A. OTHMANE, *Le protocole Mime*, Travail réalisé dans le cadre du cours de « Téléinformatique, Matières Approfondies », Institut d'Informatique, FUNDP, Namur, 1996.

[GWYER96] Mike GWYER et Rosie HARVEY, *Oracle Designer 2000 WebServer Generator Technical Overview*, withe paper, Oracle, janvier 96.

[HANDLEY95] M. HANDLEY et Jon CROWCROFT, *The World Wide Web - Beneath The Surf*, UCL Press Limited, ISBN : 1-85728-435-6 PB , Londres 1995.

[HEITML97] *HeiTML Extended Interactive HTML*, http://www.h-e-i.de/, date de dernière visite du lien :août 97.

[INSIDE97] *Web Casting, l'information sur un plateau*, in Inside Internet, numéro 6, Juin 97.

[ISAKOWITZ94] T. ISAKOWITZ, Edward A. STOHR et P. BALASUBRAMANIAN, *Designing Hypermedia Applications*, Proceedings of the twenty-seventh annual Hawaii international conference on system sciences, pp 354-365, Hawaii, 1994.

[ISAKOWITZ95a] T. ISAKOWITZ, Edward A. STOHR et P. BALASUBRAMANIAN, *RMM : A Methodology for structured Hypermedia Design*, Communications of the ACM, Volume 38, pp 34-43, Août 95.

[ISAKOWITZ95b] T. ISAKOWITZ, A. DIAZ, Vanesa MAIORANA et G. GILABERT, *RMC : A Tool To Design WWW Applications*, http://is-2.stern.nyu.edu/~tisakowi/papers/www-95/rmcase/187.html, date de dernière visite du lien : juin 97.

[LANGE94] D. LANGE, *An Object-Oriented Design Method For Hypermedia Information Systems*, Proceedings of the twenty-seventh annual Hawaii international conference on system sciences, pp 366-375, Hawaii, 1994.

[PIGNEUR96] Y. PIGNEUR, *A Framework for designing new information systems*, Lausanne, 1996.

[RFC1521] N. BORENSTEIN et N FREED, *Mime (Multipurpose Internet Mail Extension) Part One : Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, RFC1521, Bellcore, Innosoft, Septembre 93.

[SCHWABE95] D. SCHWABE, G. ROSSI et S. BARBOSA, *Abstraction, Composition and Lay-Out Definition Mechanisms in OOHDM*, Electronic Proceedings of the ACM Workshop on Effective Abstractions in Multimedia, San Fransisco, Novembre 95.

[SHNEIDERMAN91] Ben SHNEIDERMAN, *Visual User Interface for Information Exploration*, Department of Computer Science, University of Maryland, Août 91.

[SHNEIDERMAN93] Ben SHNEIDERMAN et Christopher AHLBERG, *Visual Information Seeking : Tight Coupling of Dynamic Query Filtrers with Starfield Displays*, Department of Computer Science, University of Maryland, 1993.

[SHNEIDERMAN95] Ben SHNEIDERMAN, Catherine PLAISANT et Khoa DHOAN, *Query Previews in Networked Information Systems*, Department of Computer Science, University of Maryland, Octobre 1995.

[SHNEIDERMAN97] Ben SHNEIDERMAN, *Direct Manipulation for Comprehensible, Predictable and Controllable user interfaces*, Department of Computer Science, University of Maryland, ftp://ftp.cs.umd.edu/pub/hcil/Reports-Abstracts-Bibliography/97-01.html, date de dernière visite du lien : mars 97.

[SOHDM] *A Scenario-Based Object-Oriented Methodology For Building Hypermedia Applications.*

[VANDERDONCKT93] Jean VANDERDONCKT, *Révision du document concernant la dérivation de style(s) d'interaction*, document de travail du groupe Trident, FUNDP, Namur, Novembre 93.

Annexe A.
Lexique

Terme	Traduction ou synonyme	Explications
ActiveX		Collection de technologies qui permet aux composants d'un programme d'interagir avec un autre dans un environnement réseau, indépendamment du langage dans lequel ils ont été créés.
AIFF	Audio IFF	Ce format a été développé par Apple pour stocker des échantillons de haute qualité.
Applet		Programme Java s'intégrant dans une page HTML et exécuté par la machine virtuelle intégrée par exemple dans un butineur.
AU	Audio	Extension pour les fichiers audio dans le monde Unix.
AVI	Audio Video Interleave format	Format de fichier vidéo créé par Microsoft.
AWT	Abstract Windows Toolkit	Librairie d'objets Java permettant la création d'interfaces indépendantes de toute plate-forme.
Browser	voir Butineur	
Butineur		Application du monde WWW et supportant le protocole HTTP servant à la visualisation et la navigation.
CAO	Conception Assistée par Ordinateur	
CGI	Common Gatway Interface	Standard permettant d'exécuter des programmes depuis un serveur Web. Le CGI spécifie comment passer des arguments à ces programmes via une requête HTTP.
DDE	Dynamic Data Exchange	Protocole créé par Microsoft permettant à des programmes de communiquer entre eux en utilisant un modèle Client-Serveur.
DoD	Department of Defense	
Drag & Drop	Glisser-Lacher	Moyen d'interaction simulant l'action de prendre un objet et de le déposer à un endroit qui peut être différent de son origine.
EPS	Encapsulated PostScript	Format d'image.
Freeware		Programme gratuit et librement distribué surtout via Internet. (voir aussi Sharware)
FTP	File Transfer Protocol	Protocole du monde DoD de niveau 4 servant au transfert de fichiers entre machines.
GIF	Graphics Interchange Format	Standard définissant un moyen de stockage et de transmission d'informations graphiques. GIF et Graphics Interchange Format sont des marques déposées de CompuServe Incorporeted.

HTML	HyperText Markup Language	Langage permettant d'écrire des documents hypermédias utilisant le World Wide Web comme moyen de diffusion.
HTTP	HyperText Transfert Protocol	Protocole du monde DoD utilisé par les applications du WWW pour le transfert de documents hypertextes.
Image Cliquable		Image dont certaines zones définies renvoient à une URL donnée.
Image Map	voir Image Cliquable.	
IP	Internet Protocol	Protocole du monde DoD de niveau 2 servant au routage des informations.
Java		Langage de programmation orienté objet permettant la création de programmes indépendants de la plate-forme sur laquelle ils sont exécutés. Java est une marque déposée de Sun.
JavaScript		Langage de scripting développé par Netscape pour le WWW.
JDBC	Java DataBase Connectivity	Version Java des librairies de fonctions ODBC.
JDK	Java Developper Kit	Kit de développement du langage Java composé des classes d'objets de base et des programmes de compilation et d'interprétation.
JPEG	Joint Photographic Experts Group	Mécanisme de compression d'image qui tient son nom du groupe qui l'a créé.
Mailing List		Système d'abonnement par courrier électronique afin de recevoir une série de courrier se rapportant à des sujets divers.
MAP	voir Image Cliquable.	
MIDI	Musical Instrument Digital Interface.	Protocole et spécification hardware utilisés pour transmettre des notes et des informations d'effets entre des synthétiseurs, ordinateurs ou autres instruments de musique électroniques.
Mime	Multipurpose Internet Mail Extensions	Protocole du monde DoD ayant pour but de faire transiter des documents de diverses natures (notamment multimédia) par mail « standard » (ou SMTP), c'est-à-dire tel que défini par le RFC 822.
MOV	Quicktime Movies	Format de fichier vidéo créé par Apple.
MPEG	Moving Picture Expert Group	Norme de séquence de vidéo numérique.
OCX		Extension d'OLE pour Object Linking and Embedding. Contrôles personnalisés permettant l'extension infinie de l'ensemble des contrôles Microsoft Access. Des exemples de contrôles sont les « Scroll Bar Control » et « Calendar Control ».

ODBC	Open DataBase Connectivity	L'Open DataBase Connectivity est une interface créée par Microsoft afin de connecter des bases de données hétérogènes via un même programme.
OpenDoc		Architecture de documents capables de contenir différents éléments pouvant également provenir de programmes différents.
Plugin		Extension du butineur Navigator de Netscape. Ces plugins permettent par exemple de visualiser de nouveaux formats d'images.
PPT		Format de fichier de présentation Power Point.
PPZ		Format de fichier d'animation Power Point.
QuickTime	voir MOV	
RA	Real Audio	Format de fichier son à transfert en temps réel via le Web.
RFC	Request For Comments	
SQL	Structured Query Language	Langage fournissant une interface utilisateur pour les systèmes de gestion de base de données relationnelle.
Sharware		Programme librement distribué surtout via Internet, gratuit le plus souvent pour une période d'essai donnée. Au terme de cette période, l'utilisateur a le choix d'effacer la copie ou de payer un modique contribution à l'auteur s'il désire continuer à l'utiliser. (voir aussi Freeware)
Tag		Elément du langage de description de pages HTML décrivant les entités de celles-ci.
TCP	Transport Control Protocol	Protocole du monde DoD de niveau 3 orienté mode connecté, s'occupant de rendre fiable le transport d'informations.
TGA	TarGA File Format	Format d'image.
TIFF	Tag Image File Format	Format d'image spécialement prévu pour les images scanées.
UDP	User Datagram Protocol	Protocole du monde Dod de niveau 3 orienté mode non-connecté.
URL	Uniform Ressources Locator	« Adresse » référençant un et un seul document dans le monde du WWW.
VBScript		Langage de script pour le WWW développé par Microsoft se basant sur Visual Basic.
VRML	Virtual Reality Modeling Language	Langage de description de mondes virtuels dont les éléments peuvent être reliés à Internet par des liens interactifs.
WAV		Format de fichier son créé par Microsoft.
Web	voir World Wide Web.	

World Wide Web	Toile d'araignée planétaire	Application d'Internet utilisant le protocole HTTP pour la diffusion de documents écrits en HTML.
WWW	voir World Wide Web.	
WYSIWYG	What You See Is What You Get	Mode de présentation où le mode de représentation pris en compte est l'équivalent visuel du monde réel.
XBM	X BitMap Format	Format d'image, plus connu du monde Unix.

Annexe B.

Types et sous-types Mime

Source http://www.isi.edu

Type	Subtype	Reference
text	plain	[RFC1521,Borenstein]
	richtext	[RFC1521,Borenstein]
	enriched	[RFC1896]
	tab-separated-values	[Paul Lindner]
	html	[RFC1866]
	sgml	[RFC1874]
	vnd.latex-z	[Lubos]
multipart	mixed	[RFC1521,Borenstein]
	alternative	[RFC1521,Borenstein]
	digest	[RFC1521,Borenstein]
	parallel	[RFC1521,Borenstein]
	appledouble	[MacMime,Patrik Faltstrom]
	header-set	[Dave Crocker]
	form-data	[RFC1867]
	related	[RFC1872]
	report	[RFC1892]
	voice-message	[RFC1911]
	signed	[RFC1847]
	encrypted	[RFC1847]
message	rfc822	[RFC1521,Borenstein]
	partial	[RFC1521,Borenstein]
	external-body	[RFC1521,Borenstein]
	news	[RFC 1036, Henry Spencer]
	http	[RFC1945,Berners-lee]
application	octet-stream	[RFC1521,Borenstein]
	postscript	[RFC1521,Borenstein]
	oda	[RFC1521,Borenstein]
	atomicmail	[atomicmail,Borenstein]
	andrew-inset	[andrew-inset,Borenstein]
	slate	[slate,terry crowley]
	wita	[Wang Info Transfer,Larry Campbell]
	dec-dx	[Digital Doc Trans, Larry Campbell]
	dca-rft	[IBM Doc Content Arch, Larry Campbell]
	activemessage	[Ehud Shapiro]
	rtf	[Paul Lindner]
	applefile	[MacMime,Patrik Faltstrom]
	mac-binhex40	[MacMime,Patrik Faltstrom]
	news-message-id	[RFC1036, Henry Spencer]
	news-transmission	[RFC1036, Henry Spencer]
	wordperfect5.1	[Paul Lindner]
	pdf	[Paul Lindner]
	zip	[Paul Lindner]
	macwriteii	[Paul Lindner]
	msword	[Paul Lindner]
	remote-printing	[RFC1486,Rose]
	mathematica	[Van Nostern]
	cybercash	[Eastlake]
	commonground	[Glazer]
	iges	[Parks]
	riscos	[Smith]
	eshop	[Katz]
	x400-bp	[RFC1494]
	sgml	[RFC1874]
	cals-1840	[RFC1895]
	pgp-encrypted	[RFC2015]
	pgp-signature	[RFC2015]
	pgp-keys	[RFC2015]
	vnd.framemaker	[Wexler]

	vnd.mif	[Wexler]
	vnd.ms-excel	[Gill]
	vnd.ms-powerpoint	[Gill]
	vnd.ms-project	[Gill]
	vnd.ms-works	[Gill]
	vnd.ms-tnef	[Gill]
	vnd.svd	[Becker]
	vnd.music-niff	[Butler]
	vnd.ms-artgalry	[Slawson]
	vnd.truedoc	[Chase]
	vnd.koan	[Cole]
	vnd.street-stream	[Levitt]
	vnd.fdf	[Zilles]
	set-payment-initiation	[Korver]
	set-payment	[Korver]
	set-registration-initiation	[Korver]
	set-registration	[Korver]
	vnd.seemail	[Webb]
	vnd.businessobjects	[Imoucha]
	vnd.meridian-slingshot	[Wedel]
	vnd.xara	[Matthewman]
	sgml-open-catalog	[Grosso]
	vnd.rapid	[Szekely]
	vnd.enliven	[Santinelli]
	vnd.japannet-registration-wakeup	[Fujii]
	vnd.japannet-verification-wakeup	[Fujii]
	vnd.japannet-payment-wakeup	[Fujii]
	vnd.japannet-directory-service	[Fujii]
image	jpeg	[RFC1521,Borenstein]
	gif	[RFC1521,Borenstein]
	ief	[RFC1314]
	g3fax	[RFC1494]
	tiff	[Rose]
	cgm	[Francis]
	naplps	[Ferber]
	vnd.dwg	[Moline]
	vnd.svf	[Moline]
	vnd.dxf	[Moline]
	png	[Randers-Pehrson]
	vnd.fpx	[Spencer]
	vnd.net-fpx	[Spencer]
audio	basic	[RFC1521,Borenstein]
	32kadpcm	[RFC1911]
	vnd.qcelp	[Lundblade]
video	mpeg	[RFC1521,Borenstein]
	quicktime	[Paul Lindner]
	vnd.vivo	[Wolfe]

REFERENCES

[MacMime] Work in Progress.

[RFC1036] Horton, M., and R. Adams, "Standard for Interchange of USENET Messages", RFC 1036, AT&T Bell Laboratories, Center for Seismic Studies, December 1987.

[RFC1494] Alvestrand, H., and S. Thompson, "Equivalences between 1988 X.400 and RFC-822 Message Bodies", RFC 1494, SINTEF DELAB, Soft*Switch, Inc., August 1993.

[RFC1521] Borenstien, N., and N. Freed, "MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1521, Bellcore, Innosoft, September 1993.

[RFC1563] Borenstien, N., "The text/enriched MIME content-type". RFC

1563, Bellcore, January 1994.
[RFC1866] Berners-Lee, T., and D. Connolly, "Hypertext Markup Language - 2.0", RFC 1866, MIT/W3C, November 1995.
[RFC1867] Nebel, E., L. Masinter, "Form-based File Upload in HTML", RFC 1867, Xerox Corporation, November 1995.
[RFC1872] Levinson, E., "The MIME Multipart/Related Content-type", RFC 1872, Accurate Information Systems, Inc. December 1995.
[RFC1873] Levinson, E., "Message/External-Body Content-ID Access Type", RFC 1873, Accurate Information Systems, Inc. December 1995.
[RFC1874] Levinson, E., "SGML Media Types", RFC 1874, Accurate Information Systems, Inc. December 1995.
[RFC1895] Levinson, E., "The Application/CALS-1840 Content Type", RFC 1895, Accurate Information Systems, February 1996.
[RFC1896] Resnick, P., and A. Walker, "The Text/Enriched MIME Content Type", RFC 1896, Qualcomm, Intercon, February 1996.
[RFC1911] Vaudreuil, G., "Voice Profile for Internet Mail", RFC 1911, Octel Network Services, February 1996.
[RFC1945] Berners-Lee, Y., R. Feilding, and H.Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945. MIT/LCS, UC Irvine, MIT/LCS, May 1996.

PEOPLE

[Becker] Scott Becker, <dataware@compumedia.com>, April 1996.
[Berners-Lee] Tim Berners-Lee, <timbl@w3.org>, May 1996.
[Borenstein] Nathaniel Borenstein, <NSB@bellcore.com>, April 1994.
[Butler] Tim Butler, <tim@its.bldrdoc.gov>, April 1996.
[Larry Campbell]
[Chase] Brad Chase, <brad_chase@bitstream.com>, May 1996.
[Cole] Pete Cole, <pcole@sseyod.demon.co.uk>, June 1996.
[Dave Crocker] Dave Crocker <dcrocker@mordor.stanford.edu>
[Terry Crowley]
[Eastlake] Donald E. Eastlake 3rd, dee@cybercash.com, April 1995.
[Faltstrom] Patrik Faltstrom <paf@nada.kth.se>
[Francis] Alan Francis, A.H.Francis@open.ac.uk, December 1995.
[Fujii] Kiyofusa Fujii <kfujii@japannet.or.jp>
[Gill] Sukvinder S. Gill, <sukvg@microsoft.com>, April 1996.
[Glazer] David Glazer, <dglazer@best.com>, April 1995.
[Imoucha] Philippe Imoucha <pimoucha@businessobjects.com>, October 1996.
[Katz] Steve Katz, <skatz@eshop.com>, June 1995.
[Korver] Brian Korver <briank@terisa.com>, October 1996.
[Levitt] Glenn Levitt <streetd1@ix.netcom.com>, October 1996.
[Lubos] Mikusiak Lubos <lmikusia@blava-s.bratisla.ingr.com>, October 1996.
[Lundblade] Laurence Lundblade <lgl@qualcomm.com>, October 1996.
[Matthewman] David Matthewman <david@xara.com>, October 1996.
[Moline] Jodi Moline, <jodim@softsource.com>, Aprol 1996.
[Paul Lindner]
[Parks] Curtis Parks, <parks@eeel.nist.gov>, April 1995.
[Randers-Pehrson] Glenn Randers-Pehrson <glennrp@ARL.MIL>, October 1996.
[Rose] Marshall Rose, <mrose@dbc.mtview.ca.us>, April 1995.
[Santinelli] Paul Santinelli, Jr. <psantinelli@narrative.com>, October 1996.
[Shapiro] Ehud Shapiro
[Slawson] Dean Slawson, <deansl@microsoft.com>, May 1996.
[Smith] Nick Smith, <nas@ant.co.uk>, June 1995.
[Spencer] Marc Douglas Spencer <marcs@itc.kodak.com>, October 1996.
[Henry Spencer]
[Szekely] Etay Szekely <etay@emultek.co.il>, October 1996.
[Webb] Steve Webb <steve@wynde.com>, October 1996.
[Wedel] Eric Wedel <ewedel@meridian-data.com>, October 1996.
[Wexler] Mike Wexler, <mwexler@frame.com>, April 1996.
[Wolfe] John Wolfe, <John_Wolfe.VIVO@vivo.com>, April 1996.
[Van Nostern] Gene C. Van Nostern <gene@wri.com>, February 1995.
[Zilles] Steve Zilles <szilles@adobe.com>, October 1996.

Annexe C. Outils d'édition ou de génération de pages HTML

C.1 Liste d'éditeurs HTML (septembre 96)

Nom du Software	Version	Adresse
RobEdit	1.0 beta4	http://www.ozemail.com.au/~html/
SpiderPad	1.1.2	http://www.sixlegs.com/
SuperPad		
Swag-Man	1.0	http://www.iinet.net.au/~bwh/swagman.html
TC Director	2.01 b23	http://www.linkstar.com/page/tashcom-software/
TextPad32	1.28	http://WWW.NetEx.NET:80/w95/windows95/utils/
The Web Media Publisher	2.6	http://www.wbmedia.com/software.html
THEdit		ftp://iworks.ecn.uiowa.edu/pub/comp.hp/
Visual HTML++	1.0	http://www.nfinity.com:80/~ellussion/vhtml.htm
Visual HTMLBoard		http://www.adaptive-computer.com/vhb.html
Visual Web	??	http://www.vizweb.com/
W3e	3.01a	http://www.nce.ufrj.br/~cracky
Web Designer	1.0.38	http://www.cybercity.dk/users/ccc2277/webdes.html
Web Ed	1.1.1 beta	http://www.ozemail.com.au:80/~kread/webed.html
Web Etch	.3b	http://mrcc.com/webetch03.html
Web Page Creator	4.70	http://www.geocities.com/SiliconValley/Park/2514/
Web Spinner	1a	http://www.execpc.com/~flfsoft/webspin.html
Web Weaver	5.1	http://www.tiac.net/users/mmm/webweav.html
Webber	1.4	http://www.csdcorp.com/
Webedit	1.4cbeta2	http://www.nesbitt.com/
Webedit 32Pro & Standard	2.0 beta2	http://www.nesbitt.com/
WebElite	.99c	http://www.safety.net/webelite/
WebMaker	4.0	http://www.bcpl.lib.md.us/~vmclean/ocean.html
Webmania	1.5	http://www.q-d.com
WebMate		http://www.cerritos.edu/~lebro002/
WebPen	1.0	http://www.execpc.com/~infothek/webpen.html
WebPen Pro	??	http://www.execpc.com/~infothek/webpen.html
WebText	1.0	http://www.pacificrim.net/~proactiv/webtext/
WebThing	2.54	http://id.mind.net/~lutusp/webthing.htm
Webwizard	1.1	http://www.halcyon.com/webwizard
WebWord	1.0 b	http://jumper.mcc.ac.uk/~careyb/webword/webword.htm
Windows	1.1	ftp://ftp.eznet.net/pub/win/editors/spad.zip
WorldDoc	1.01	http://www.spii.com./
Xantippe	1.0	ftp://ftp.netcom.com/pub/rl/rlai/www/info/xantinfo.html
Zerosoft Halo		http://www.ozemail.com.au/~dmoreitz/halo/

C.2 Panorama et étude d'éditeurs HTML

Il faut noter ici que nous analysons deux classes d'éditeurs HTML, à savoir les programmes commerciaux et les programmes sharewares. Cependant, il faut également savoir que les programmes appartenant à la première classe sont pour la plupart des versions d'évaluation (dit version bêta). De plus, les logiciels sont testés uniquement sur la plate-forme Windows 95.

Notons enfin que l'évolutivité de ce type d'outils est particulièrement rapide et que le lecteur pourra se référer aux URL citées pour obtenir les dernières informations sur les produits.

Nous appliquons un plan d'évaluation systématique qui permettra de mieux comparer et repérer les caractéristiques, avantages et inconvénients des différents programmes. Ensuite, nous donnons en aperçu les caractéristiques essentielles, avantages et inconvénients des programmes phares du moment.

C.2.1 Plan d'évaluation

Le plan d'évaluation se compose de différentes parties. La première consiste en une fiche d'identité et la deuxième en une fiche technique.

Nom du programme	Dénomination du programme analysé
Numéro de version	Numéro de la version étudiée
Type de programme	• Shareware si le programme est libre d'utilisation pendant une période déterminée, ou bien que seulement une partie de ces fonctions est accessible à l'utilisateur non enregistré. • Freeware si le programme est entièrement libre d'utilisation. • Commercial : si le programme est payant.
Editeur	Nom de l'éditeur (ou de l'auteur) correspondant au programme.

Tableau C-1 : Fiche d'identité du plan d'évaluation.

Type de programme	• Indépendant (stand-alone) si le programme fonctionne seul, sans avoir recours à un autre programme. • Complément si le programme est le complément d'un autre tel qu'un ensemble de macros pour un traitement de texte. • Autre si le programme fait appel à une technologie différente des possibilités ci-dessus.
Type d'affichage	• Style texte si la fenêtre principale affiche le code en mode texte. • WYSIWYG si la fenêtre principale affiche le code sous forme graphique interprétée. • Autre si le programme emploie une autre possibilité que

	celles citées ci-dessus.
Aperçu	• Oui / Non si le programme permet l'interprétation directe du code HTML ou non. • Implicite si le programme le réalise implicitement. • Butineur si le programme délègue l'affichage graphique à un butineur.
Correction de la syntaxe HTML	Oui / Non selon que le code HTML est corrigé ou non.
Correction de l'orthographe	Oui / Non selon que le programme propose la correction de l'orthographe ou non.
Butineur inclus	Oui / Non selon qu'un butineur Web est inclus ou non.
Support des images cliquables	Oui / Non selon que le programme intègre certains assistants facilitant la création d'images cliquables.
Support d'images Gif transparentes	Oui / Non selon que le programme intègre certains assistants facilitant la création d'images Gif transparentes.
Support de tableaux	Oui / Non selon que le programme intègre certains assistants facilitant la création de tableaux
Support de formulaires	Oui / Non selon que le programme intègre certains assistants facilitant la création des formulaires.
Support des cadres (Frames)	Oui / Non selon que le programme intègre certains assistants facilitant la création des cadres (Frames)
Personnalisation des Tags	Oui / Non si le programme propose de personnaliser certains Tags
Support des scripts	Oui / Non (CGI - C - Java - PL)
Support des extensions	• HTML 3.0 si les extensions HTML 3.0 sont supportées et gérées correctement. • Netscape 2.0 et plus si les extensions Netscape ou plus sont supportées et gérées correctement. • MS Explorer 2.0 et plus si les extensions Ms Explorer ou plus sont supportées et gérées correctement.
Importation d'autres formats de fichier	Tels que Word, RTF, ASCII, TXT...

Tableau C-2 : Fiche technique du plan d'évaluation

C.2.2 Evaluation

Nom	FrontPage Editor	WebMania	HotDog Pro	Dida Pro	The Web Media Publisher	Claris Home Page
Version	97	1.2	3	2.20	2.4	1
Type de programme	Commercial	Shareware	Commercial	Shareware	Shareware	Commercial
Editeur	Microsoft	Q&D Software Development	Sausage Software	Godfrey Ko	Web Media Inc.	Claris Corporation
Type	Stand Alone	Stand Alone	Stand Alone	Stand Alone	Stand Alone	Stand Alone
Affichage	WYSIWYG	Texte	Texte	Texte	Texte	WYSIWYG
Aperçu	Oui	Non	Butineur	Oui	Butineur	Oui
Correction HTML	Oui	Non	Oui	Non	Non	Oui
Correction Ortho.	Oui	Non	Oui	Non	Non	
Butineur	Oui	Non	Oui	Non	Non	Oui (Local)
Maps	Oui	Non	Non	Non	Non	Oui
Gifs Transp.	Oui	Non	Non	Oui	Non	Oui
Tables	Oui	Oui	Oui	Oui	Oui	Oui
Forms	Non	Oui (!!)	Oui	Oui	Oui	Oui
Frames	Oui	Non		Non	Oui	Oui
Tags	Non	Oui	Oui	Oui	Non	Oui
Scripts	CGI	Non	CGI, C, PL	Non	Non	Java
HTML 3.0	Oui	Non	Oui	Non	Oui	Non
Netscape	Oui	Non	Oui	Non	Oui	Non
Explorer	Oui	Non	Oui	Non	Oui	Non
Importations	Txt, Rtf	Txt	Txt, Tpl, ...	Txt	Txt	Txt

Tableau C-3 : Tableau d'évaluation, première partie.

Nom	Netscape Composer	HTMLEd32	Web Edit	PageMill	Navi Press
Version	4	1.5	2.0		1.1
Type de programme	Commercial	Shareware, commercial pour la version Pro	Shareware	Commercial	Commercial
Editeur	Netscape Corp	Internet Software Technologies	KnowledgeWorks Inc.	Adobe	NaviSoft
Type	Stand Alone	Stand Alone	Stand Alone	Stand Alone	Stand Alone
Affichage	WYSIWYG	Texte couleur	Texte	WYSIWYG	WYSIWYG
Aperçu	Butineur	Butineur	Oui	Implicite	Implicite
Correction HTML	Oui	Non (Pro)	Non	Oui	Non
Correction Ortho..	Oui	Non	Oui	Oui	Non
Butineur	Oui	Non	Non	Non	Non
Maps	Non	Non	Oui	Oui	Non
Gifs Transp.	Oui	Non	Non	Oui	Non
Tables	Oui	Oui	Oui	Oui	Oui
Forms	Oui	Non (Pro)	Non	Oui	Oui
Frames	Non	Non	Oui	Oui	Non
Tags	Non	Oui	Oui	Non	Oui
Scripts	Oui	Non	Non	Oui	Non
HTML 3.0	Oui	Oui	Oui	Oui	Oui
Netscape	Oui	Non	Oui	Oui	Non
Explorer	Non	Non	Oui	Oui	Non
Importations	Txt	Txt, (Rtf Pro)	Non	Txt, Doc, Wps, Xls, Dbf, …	Non

Tableau C-4 Tableau d'évaluation, deuxième partie.

C.2.3 Les logiciels phares

C.2.3.1 HotDog Pro 3.0 - Sausage Software

Aperçu. Cet éditeur est conçu pour la plate-forme MS-Windows en un programme indépendant orienté « texte ». Il supporte la vérification de la syntaxe HTML sur demande.

En voici une copie d'écran :

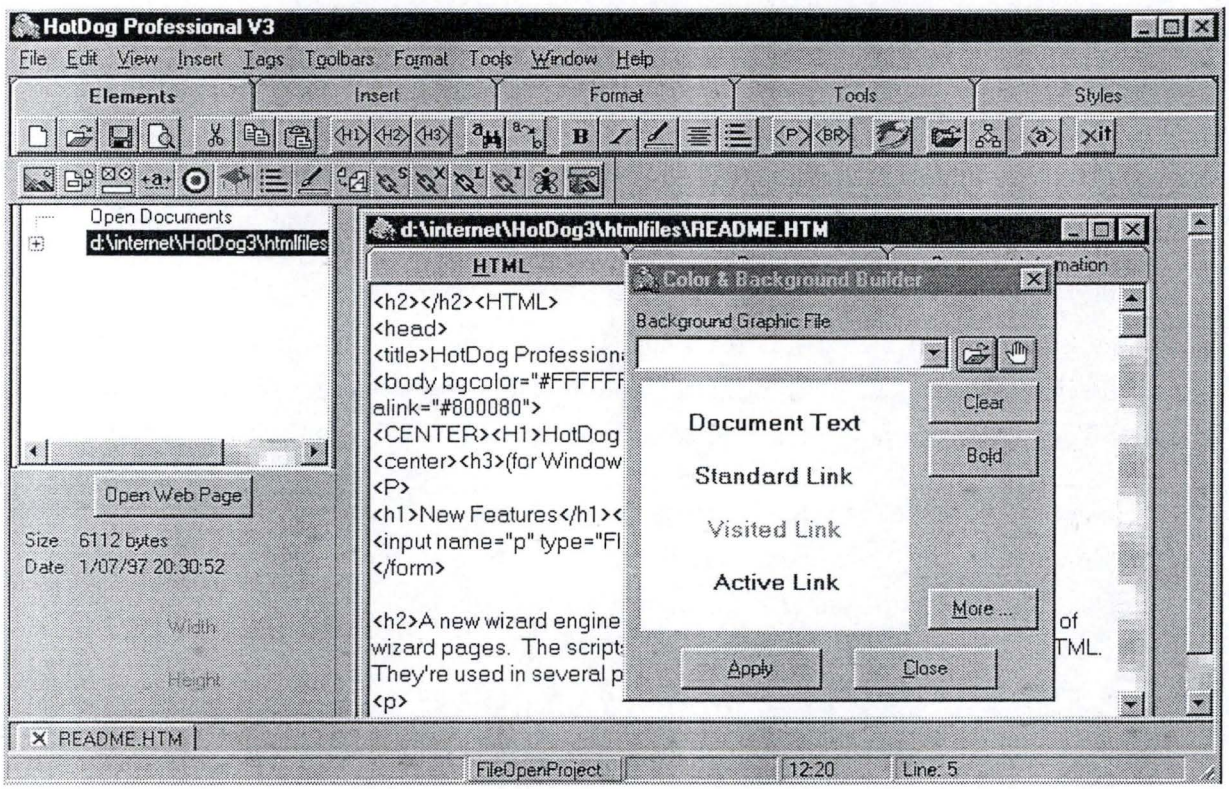


Figure C-1 : Capture d'écran de HotDog Pro 3.0

Aperçu des pages intégré. Le programme Rover est intégré afin de fournir une fenêtre d'aperçu au sein même de l'éditeur (sans l'utilisation d'un butineur externe). Ceci est particulièrement utile pour vérifier, entre autres, le format des tableaux.

Insertion d'images et de liens. En général l'insertion d'images et de liens se réalise aisément. Un gestionnaire de fichiers est présent, ce qui permet à l'utilisateur de sélectionner un fichier et d'insérer le lien qui lui correspond dans le document.

Gestion des tableaux et des formulaires. La présence d'un éditeur de tableaux améliore sensiblement l'utilité du programme. En effet, il est possible de définir la taille et les autres propriétés du tableau et de les insérer ensuite dans la feuille d'édition. L'éditeur permet également de rééditer des informations préalablement choisies.

Facilités d'utilisation. L'utilisateur est constamment guidé lors de l'insertion de liens (le type de lien et l'information correspondante sont demandés) et de listes. Une aide présentant les caractéristiques du langage HTML est présente, ce qui est un avantage pour l'utilisateur novice. La possibilité de « colorier » les Tags HTLM rend la lecture du texte plus aisée.

Préparation de pages pour les serveurs. HotDog se charge aisément de la transformation des noms de fichier (.htm en .html), du formatage de texte (Dos vers UNIX), de la présence de l'heure et de la date et d'autres tâches associées à la présence d'un document sur un serveur Web.

Support d'HTML 3.0, de Netscape 1.X et de Internet Explorer. La plupart des Tags HTML 3.0 et Netscape sont supportés ainsi que ceux concernant Internet Explorer. De plus, des assistants sont inclus pour aider l'utilisateur à créer, par exemple, des Gifs animés, ou insérer une vidéo. De même, HotDog Pro supporte les feuilles de style, fait assez rare pour ne pas le signaler.

Les inconvénients d'HotDog. Un peu lourd graphiquement, il n'est pas aisé, d'un premier coup d'œil, de retrouver les informations pertinentes ; mais on peut signaler l'effort déjà réalisé à ce niveau par rapport à la version précédente.

Conclusion. Les nouvelles possibilités, telles que l'aperçu de la page, l'édition de tableaux ou la vérification de la syntaxe HTML, mettent en avant ce programme orienté texte. Toutefois, le service utilisateur d'HotDog n'est pas à la hauteur du produit mais HotDog contient vraiment toutes les fonctions désirées et est digne d'un bon éditeur HTML.

C.2.3.2 HoTMetaL PRO 3.0 - SoftQuad

Aperçu. HotMetla Pro 3.0 est implémenté pour les plate-formes Windows 3.x/NT/95 (3.0) , Unix et Mac (2.0). C'est un programme indépendant proposant une interface hybride et orienté texte et graphique. Il propose la vérification de la syntaxe HTML pendant l'édition et l'importation de document.

En voici une copie d'écran :

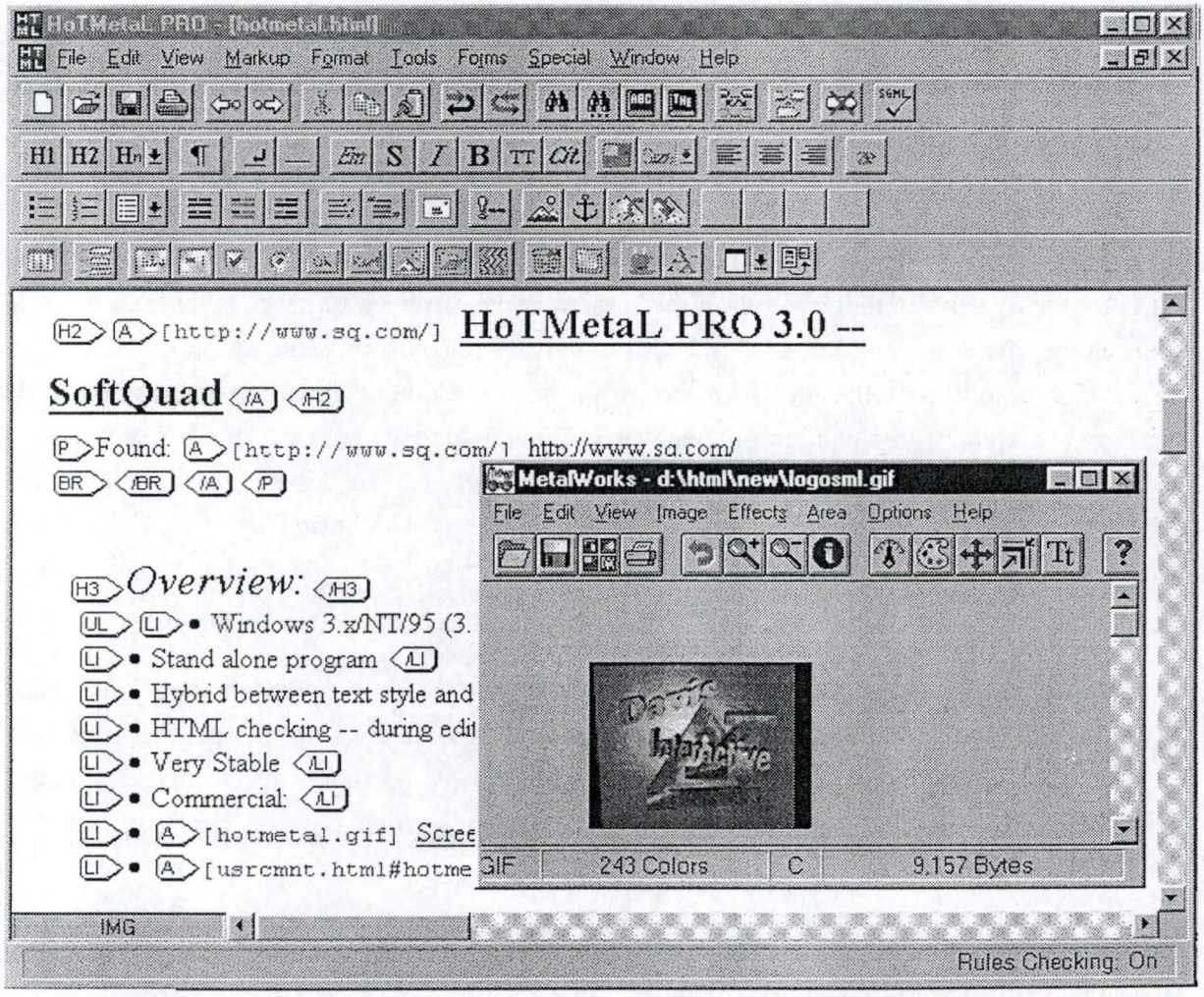


Figure C-2 : Capture d'écran de HotMetal Pro 3.0

Large éventail d'options concernant l'importation de fichiers. Le programme peut importer un grand nombre de formats de fichiers et tenter de créer un fichier HTML à partir de ceux-ci (on retrouve entre autres les format Word, Amipro et Wordperfect).

Facilité de lecture du document. Les Tags sont représentés sous forme d'icônes, ce qui simplifie la lecture du document. En effet, lorsque l'on travaille avec un document contenant beaucoup de sections similaires on s'aperçoit facilement des avantages de cette fonction.

Les tableaux et les formulaires. Il est très facile d'insérer des tableaux pouvant être réédités par la suite. Leur création s'effectue de la même manière que les outils présents dans un traitement de texte. La gestion des formulaires est également très aisée. En effet, depuis le version 3.0, un éditeur graphique est incorporé, ce qui permet de se rendre compte immédiatement du résultat obtenu.

L'insertion de Tags. Lorsque l'on sélectionne un élément à insérer ou que l'on clique sur la barre de boutons, le Tag (avec ses attributs par défaut) correspondant est inséré dans la fenêtre d'édition. A ce moment, un menu permet de choisir les attributs de ce Tag dès que le curseur se positionne sur lui.

Un éditeur orienté texte. Ce programme ressemble beaucoup à un éditeur de texte dont le formatage apparaît à l'écran. Il est aussi suffisamment « intelligent » pour simplifier quelques opérations telle que l'insertion d'un nouvel élément dans une liste (il suffit de passer à la ligne après chaque élément afin d'en rajouter un autre). Dans un programme orienté texte traditionnel, l'auteur peut par exemple effacer ou déplacer accidentellement certains Tags, mais ceci est impossible avec HotMetal. Un option du programme peut forcer l'utilisateur à respecter la syntaxe d'HTML et permet donc de garder des documents tout à fait corrects.

Editeur d'image intégré. L'éditeur d'image SoftQuad et le programme de cataloguage Metalworks, sont maintenant inclus dans HotMetal Pro. Ceux-ci permettent à l'utilisateur de modifier rapidement les images (tant au niveau format de fichier qu'au niveau nombre de couleurs) et de leur appliquer un fond transparent ainsi que de créer des images cliquables.

Autres caractéristiques. Un enregistreur de macrocommandes est présent, ainsi qu'un correcteur orthographique et un éditeur de Frames.

Les inconvénients d'HotMetal Pro. La boîte de dialogue permettant de changer les attributs n'est pas présente sur la rangée de boutons affichés à l'écran. L'utilisateur est donc obligé de passer en permanence par les menus. De plus, cette rangée de boutons prend une place importante sur l'écran, ce qui ne facilite pas la lecture de celui-ci.

Conclusion. HoTMetaL Pro est un leader dans ce type de marché, et malgré l'impossibilité de créer de nouveaux Tags, l'utilisateur averti sera content d'avoir un document respectant exactement la syntaxe HTML. C'est donc un éditeur puissant, assez complet, permettant facilement de changer les attributs des Tags, de créer de longs documents et de gérer efficacement les images.

C.2.3.3 Claris Home Page - Claris Corporation[1]

Aperçu. Claris Home Page est implémenté pour les plate-formes Windows NT/95 et Mac OS. Le programme est indépendant et de type WYSIWYG.

En voici une copie d'écran :

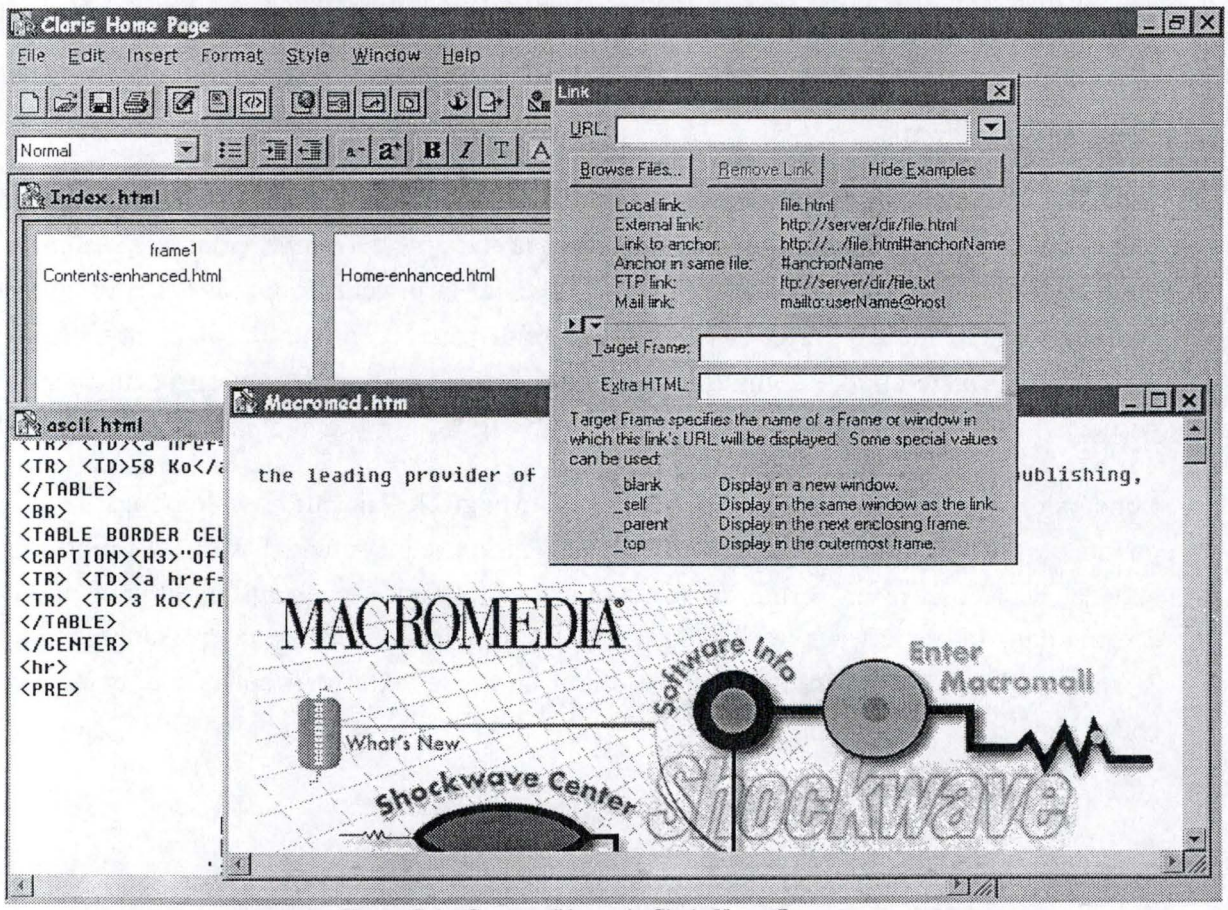


Figure C-3 : Capture d'écran de Claris Home Page

[1] Claris Home Page est un produit Claris Corporation http://www.claris.com/

Edition WYSIWYG. Cet éditeur est entièrement WYSIWYG. Ce qui ne l'empêche pas d'avoir un mode d'édition texte pour corriger le code HTML manuellement Un mode de pré-visualisation total existe également (afin d'enlever les caractères représentant les retours de chariots par exemple) et l'appel à un butineur externe permet de contrôler également la page pour un système particulier. En effet, les Tags non reconnus par Claris Home Page sont affichés en rouge.

Edition de tables, formulaires et Frames. L'édition des tables, formulaires et Frames est supportée de façon intuitive. La mise en place des tables est simple. Par exemple, pour agrandir une colonne il suffit d'étirer celle-ci en cliquant sur son bord et en le tirant vers la droite ou la gauche. Un mode d'édition des Frames permet de créer et de positionner celles-ci dans une page principale.

Editeur graphique. L'édition des graphiques est également prise en compte. En effet, ce module permet l'édition des images, de rendre une couleur transparente et de créer des images cliquables.

Quelques inconvénients. Dans cette version d'évaluation, certains Tags n'étaient pas encore reconnus. Il est cependant possible que dans la version définitive cela soit résolu. Les tableaux ont aussi tendance à prendre d'office toute la largeur de la page. La création de formulaires se résume à placer des éléments sur une page. L'éditeur a également tendance à considérer la page entière comme le formulaire, alors qu'il ne devrait en occuper qu'une partie.

Conclusion. Logiciel très intuitif et très visuel. Malgré le fait que tous les Tags ne soient pas reconnus, la création de pages HTML devient un jeu d'enfant. Il suffit d'introduire le texte et de le mettre en forme en sélectionnant une phrase et en lui appliquant un style comme dans la plupart des traitements de textes. Cependant, malgré la puissance du mode WYSIWYG, il n'est pas rare de devoir passer en mode texte pour corriger directement le code HTML.

C.2.3.4 Netscape Composer - Netscape Corp.

Aperçu. Netscape Composer est implémenté pour la plate-forme MS-Windows 95/NT. Il s'agit d'un programme faisant partie de la suite Netscape Communicator. Il est WYSIWYG et possède une option de vérification de la syntaxe HTML pendant l'édition et l'importation.

En voici une copie d'écran :

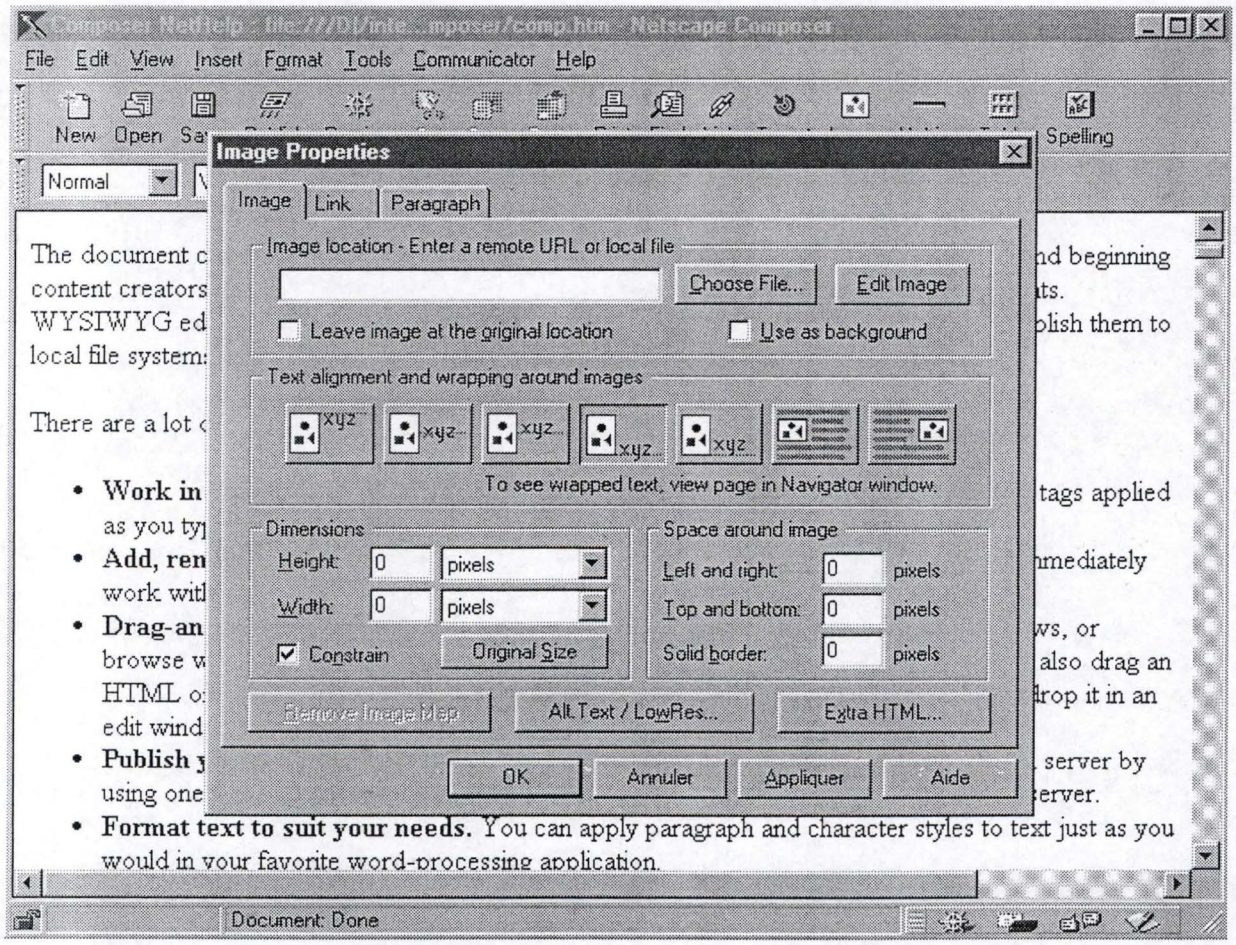


Figure C-4 : Capture d'écran de Netscape Composer

Visualisation des documents par le butineur. Ce programme fait partie de la suite Netscape Communicator. C'est en fait un butineur incorporant des possibilités d'édition. L'approche est donc assez intuitive et la création de documents se réalise facilement et rapidement. Les utilisateurs novices du langage HTML y trouveront ce qu'ils souhaitent.

Gestion des images. La gestion des images est bien réalisée et on peut y trouver entre autres tous les attributs désirés ainsi que des aperçus de la façon dont la commande ALIGN affecte l'image.

Les inconvénients de Netscape Composer. Comme pour les versions précédentes qui portaient le nom de Netscape Navigator Gold, la gestion des Frames n'est pas présente. Ce qui est dommage d'autant plus que c'est la société Netscape elle-même qui les a inventés. Sinon, de nombreux efforts ont été réalisés par exemple du coté des attributs des Tags.

Conclusion. Malgré les quelques défauts, ce logiciel est simple d'emploi, intuitif et très proche du navigateur de la suite. Du fait que ce programme est écrit par Netscape, il n'est pas capable de gérer les extensions HTML de Internet Explorer.

C.2.3.5 PageMill 2.0 - Adobe[2]

Apercu. PageMill est implémenté pour les plate-formes MS-Windows et Mac. Il consiste en un programme indépendant de type WYSIWYG possédant l'option de vérification de la syntaxe HTML sur demande.

Caractéristiques principales. PageMill est complètement WYSIWYG, le drag & drop est supporté pour l'importation d'images, de sons, le déplacement de textes, de Tags, la création de liens. Les tableaux sont créés et présentés graphiquement à l'utilisateur ainsi que les Frames et les formulaires. La disposition du texte autour d'une image est aussi permise. Des pages complexes offrant une bonne organisation de l'information peuvent ainsi être créées.

Les possibilités multimédias. PageMill autorise l'emploi des mêmes Plugins que le navigateur de Netscape, ce qui permet de présenter des pages incluant entre autres des fichiers Acrobat (PDF), QuickTime, ShockWave. Le support des Applets Java est présent.

Conversation des formats de fichiers. Le programme convertit automatiquement les fichiers son par exemple au format AIFF, SND, WAV et les images au format Internet (comme gif ou jpeg), il n'est donc plus nécessaire de disposer d'un autre programme pour cette tâche. Les images Gif animées sont également supportées et présentées à l'utilisateur lors de la pré-visualisation du document (aucun Plugin spécifique n'est donc nécessaire, ce qui permet de créer facilement des logos animés). Une large librairie de filtres de conversion est présente, incluant les formats de fichier des traitements de textes et les bases de données les plus populaires. Il est donc possible d'importer directement des fichiers Microsoft Word, Wordperfect, Excel ou Dbase.

Le côté traitement de texte. Il est possible de visualiser le code HTML qui est mis en page automatiquement par PageMill. La lecture du code est donc facilitée et aucun autre programme traitant du texte n'est nécessaire afin d'effectuer une correction. Lors de l'édition en mode texte certains Tags peuvent être cachés afin d'améliorer la lecture et le choix de couleur s'effectue par l'appel d'une palette graphique (inutile donc de connaître le code d'une couleur). Les fonctions rechercher/remplacer du traitement de texte sont aussi disponibles.

---

[2] PageMill est un produit Adobe http://www.adobe.com/proindex/pagemill/

Gestion des Frames. La création des Frames est très simple et intuitive. Il suffit d'aller chercher une « ligne » sur le bord de l'écran et de la tirer jusqu'à la position qui déterminera le bord d'un nouveau cadre. Cependant, si la création est aisée, la gestion des liens entre Frames d'une même page est complexe car elle nécessite de passer par plusieurs menus.

C.3 Liste d'outils de publication de base de données pour le Web

Source Internet World http://www.iw.com/1997/06/dealing.html. Compilation du 03/03/97 par Nelson H. King.

C/S = Client/Server - ADE = Application Development Environment

Companie	Produit	Version	Type	Notes	OS Support	URL
Dynasty Corporation	Open Internet Series		C/S ADE, Web Extensions	Partitioning specialist	Unix, NT	http://www.dynasty.com/
Microsoft Corporation	Visual FoxPro	5.0	C/S ADE, Web extensions	ODBC	Win95, NT	http://www.microsoft.com/
Microsoft Corporation	Visual Basic	5.0	C/S ADE, Web extensions	ActiveX development, VBScript	Win95, NT	http://www.microsoft.com/
Oracle Corporation	Developer/2000		C/S ADE, Web extensions		Unix, Win95, NT	http://www.oracle.com/
Powersoft	PowerBuilder	5.0	C/S ADE, Web extensions	See also Sybase	Win95, NT	http://www.powersoft.com/
Prolifics Corporation	JAM Web		C/S ADE, Web Extensions	formerly JYACC	Unix, Win95, NT	http://www.prolifics.com/
Unify Corporation	Unify Web		C/S ADE, Web Extensions	Partitioning specialist	Unix, NT	http://www.unify.com/
Information Builders, Inc.	Cactus for the Web	2.0	C/S ADE, Web Extensions		MVS, Unix, Win95, NT	http://www.ibi.com/special/cactus/web/
Centura Software, Inc.	Centura Web Developer	?	C/S ADE, Web Extensions	frm.Gupta. Req. Foresight server	Unix, Win95, NT	http://www.centurasoft.com/products/development/web_

						developer/
askSam	askSam Professional	3.0	Desktop DB to Web	Data collection only	Win95, NT	http://www.asksam.com/
Claris Software, Inc.	FileMaker Pro	3.0	Desktop DB to Web	Also server	Mac, Win95, NT	http://www.claris.com/
Microsoft Corporation	Access	5.0	Desktop DB to Web	ODBC	Win95, NT	http://www.microsoft.com/
AimTech Corporation	Jamba	1.0	Java ADE			http://www.jamba.com/
Apptivity Corporation	Apptivity	1.0	Java ADE	Also server. JDBC	Win95, NT	http://www.apptivity.com/
Asymetrix Corporation	SuperCede	1.0	Java ADE	JDBC	Win95, NT	http://www.asymetrix.com/
BulletProof Corporation	Jdesigner		Java ADE		Win95, NT	http://www.bulletproof.com/
Microsoft Corporation	Visual J++	1.1	Java ADE	ODBC	Win95, NT	http://www.microsoft.com/
Netiva Software, Inc.	Netiva	beta	Java ADE	Simple DB	Win95, NT	http://www.netiva.com/
Penumbra Software, Inc.	Mojo	2.0	Java ADE	JDBC, also enterprise version	Win95, NT	http://www.penumbrasoftware.com/
Sun Microsystems, Inc., Sunsoft	Java WorkShop	1.0	Java ADE	JDBC	Unix, Win95, NT	http://java.sun.com/
Symantec	Visual Café Pro	2.0	Java ADE	dbAnywhere	Win95, NT	http://www.symantec.com/
Object Design, Inc.	ObjectStore/Web, PSE Pro	4.0.2	ODBMS, Java ADE		Unix, Win95, NT	http://www.odi.com/
GemStone Systems, Inc.	GemStone, Web Server	5.0	ODBMS, Web ADE	ParcPlace Visual Wave	Unix, Win95, NT	http://www.gemstone.com/
O2 Technologies, Inc.	O2	5.0	ODBMS, Web	C++	Unix, Win95, NT	http://www.o2tech.

			Extensions			com/
Versant Object Technology	Versant	5.0	ODBMS, Web Extensions		Unix, Win95, NT	http://www.versant.com/
Informix Corporation	Universal Server, Web Datablade	1.1	ORDBMS, Web Extensions	Illustra	Unix, Win95, NT	http://www.informix.com/
IBM	DB2, WWW Connection		RDBMS, Web Extensions			http://www.ibm.com/
Microsoft Corporation	SQL Server	6.5	RDBMS, Web Extensions		NT	http://www.microsoft.com/
Sybase Corporation	SQL Server, Net Impact	11.0	RDBMS, Web Extensions	also jDBC Connect	Unix, Win95, NT	http://www.sybase.com/
Level 5 Corporation	Quest	1.0	Search Tool	Web DB also	Win95, NT	http://www.l5r.com/
Microsoft Corporation	Visual InterDev	1.0	Web ADE	ActiveX	Win95, NT	http://www.microsoft.com/
Open Environment Corporation	Mambo	1.0	Web ADE	Now Borland	Unix, NT	http://www.openenv.com/
SourceCraft, Inc.	IntelliCraft	beta	Web ADE	JDBC/ODBC	Win95, NT	http://www.sourcecraft.com/
Sapphire Group, Inc.	PageBlazer	2.0	Web ADE	Wireless Connect speciality	Win95, NT	http://www.pageblazer.com/
Simware, Inc.	Salvo	3.5	Web ADE	DEC partner	VMS, NT	http://www.simware.com/products/salvo/
Next Software, Inc.	WebObject Enterprise	3.0	Web ADE	CGI Scripting	Unix, Win95, NT	http://www.next.com/
Bluestone	Sapphire/Web	1.1	Web C/S ADE	C++	Unix	http://www.bluestone.com/
Borland International	IntraBuilder C/S	1.01	Web C/S ADE	Delphi technology, JavaScript	Win95, NT	http://www.borland.com/

Computer Associates, Inc.	Jasmine	1.0	Web C/S ADE	ORDBMS	Unix,NT	http://www.cai.com/products/jasmine.htm
Neuron Data, Inc.	Web Elements	2.0	Web C/S ADE	Needs Elements 2.0	Everything	http://www.neurondata.com/
Usoft, Inc.	U-Soft	1.0	Web C/S ADE	Business rules partitioning	Win95, NT	http://www.usoft.com/index.html
Powersoft, Inc.	PowerSite	alpha	Web C/S ADE	Currently NetImpact	Unix, Win95, NT	http://www.powersoft.com/products/internet/netimpct/index.html
TopTier, Inc.	TopTier:Enterprise Builder	1.0	Web DB Access	browser based	Win95, NT	???
DataRamp, Inc.	DataRamp		Web DB Connect	ODBC	Win95, NT	http://www.dataramp.com/
Intelligent Environments, Inc.	Amazon	1.1	Web DB Connect	Legacy data specialist	NT	http://www.ieinc.com/
Maximum Information, Inc.	WebC		Web DB Connect		Unix, Win95, NT	http://www.maximum.com/
MicroRim Corporation	R:Web	1.1	Web DB Connect	ODBC/R:Base	Win95, NT	http://www.microrim.com/
Microsoft Corporation	Advanced Data Connector	1.0	Web DB Connect	ODBC/IIS, formerly Aspect Corp.	NT	http://www.microsoft.com/adc/
MicroStrategy, Inc.	DSS Web	1.0	Web DB Connect	DSS connection only		http://dssweb.strategy.com/
Nomad Development Corp.	WebDBC	2.5	Web DB Connect	Similar to Cold Fusion	Mac, Solaris, Win95, NT	http://web.ndev.com/
Open Horizons, Inc.	Connection for Java		Web DB Connect	ODBC		http://www.openhorizon.com/
OpenConnect Systems,	OC://WebConnect	2.5	Web DB Connect	Legacy data specialist		http://www.oc.com

Inc.						/
XDB Corporation	JetConnect Pro	1.0	Web DB Connect	free first copy	Win95, NT	http://www.xdb.com/
Inline Information Services, Inc.	iHTML Pro	2.0	Web DB Connect	Server based	Win95, NT	http://www.ihtml.com/
Showbase Media, Inc.	Showbase	1.0	Web DB Connect	Multimedia too	Unix, Win95, NT	http://www.showbase.com/
Virtuflex Software, Inc.	Virtuflex	1.1	Web DB Connect		Unix	http://www.virtuflex.com/
Ablaze Software	Ablaze Database System	1.0	Web Publishing	Uses MS Access	Win95, NT	http://www.ccpartner.se/ads/6.html
askSam Corporation	askSam Web Publisher	1.0	Web Publishing		Win95, NT	http://www.asksam.com/
Caere Corporation	OmniForm Internet Publisher	2.01	Web Publishing		Win95, NT	http://www.caere.com/
InMagic Corporation	DB/Text	1.0	Web Publishing	Text retrieval	NT	http://www.inmagicinc.com/
Allaire Corporation	Cold Fusion	2.0	Web Server Apps	Extended HTML	Win95, NT	http://www.allaire.com/
Corel Corporation	Web.Data	1.0	Web Server Apps	ODBC	Win95, NT	http://www.corel.com/
Lotus Corporation (IBM)	Domino	4.5	Web Server Apps	Notes on the Internet	Unix, Win95, NT	http://domino.lotus.com/
Microsoft Corporation	IIS	3.0	Web Server Apps	Jscript, VBScript	Win95, NT	http://www.microsoft.com/
Netscape Corporation	Enterprise Server	3.0	Web Server Apps	JavaScript	Unix, Win95, NT	http://www.netscape.com/
Expertelligence, Inc.	WebBase	4.0	Web Server DB	Is a server	Win95, NT	http://www.webbase.com/
WebMate	WebMate	1.0	Web Server DB			http://www.webma

Technologies, Inc.						te.com/
NetObjects Corporation	NetObjects Fusion	2.0	Web Site Development	Weak data elements	Win95, NT	http://www.netobjects.com/
NetDynamics, Inc.	NetDynamics			Formerly Spider Tech.		http://www.netdynamics.com/
NetScheme Solutions, Inc.	InterMart Toolkit					??
Open Software Associates	OpenWeb					http://www.osa.com/
Rogue Wave, Inc.	Dbtools++					http://www.roguewave.com/
Transarc Corporation	Delight for Java					http://www.transarc.com/

Annexe D. Description d'autres méthodologies permettant le développement d'applications hymermédias

D.1 Object-Oriented Hypermedia Design Model (OOHDM)

Cette méthode est centrée sur quatre activités principales qui sont: le design conceptuel (conceptual design), le design navigationnel (navigational design), le design de l'interface abstraite (abstract interface design) et l'implémentation. Lors de chaque activité, un ensemble de modèles orientés objet est construit ou enrichi par l'itération précédente.

D.1.1 Le design conceptuel

Lors de cette phase, un modèle du domaine de l'application est construit en utilisant la méthode du schéma conceptuel (orienté objet). A cette méthode viennent s'ajouter certains attributs ainsi que certaines primitives. Le but principal poursuivi par cette première étape est de capter la sémantique du domaine d'une façon aussi neutre que possible (en s'occupant un minimum du type d'utilisateurs et de tâches).

D.1.2 Le design navigationnel

Dans la méthodologie OOHDM, une application est considérée comme une vue navigationnelle située au-dessus du domaine conceptuel et c'est dans cette étape que le designer considérera les types d'utilisateurs et le type de tâches qu'ils doivent effectuer. Ceci montre bien le fait que les applications hypermédias sont centrées sur la notion de navigation. Plusieurs vues d'un même domaine peuvent être exprimées par différents modèles navigationnels relatant le même schéma conceptuel. OOHDM définit un ensemble de classes navigationnelles : les noeuds, les liens et les structures d'accès[3] organisées en contextes organisationnels.

Les noeuds sont définis en combinant différents attributs (un attribut type et un attribut ancrage) des différentes classes composant le schéma conceptuel. Ces noeuds peuvent être définis de façon atomique ou composite.

Dans ce modèle, les liens représentent la réalisation navigationnelle des relations définies dans le schéma conceptuel. Des attributs de liens, de comportements, de source, de cible et de cardinalité sont également définis pour la classe de lien.

---

[3] les noeuds et les liens ont une sémantique identique a celle utilisée dans les applications hypermédia tandis que les structures d'accès représentent des alternatives aux manière d'accéder aux noeuds (indexs, "guided tour" cfr infra.)

Les structures d'accès sont comparables à des index ou à des dictionnaires et sont très utiles afin d'aider l'utilisateur final dans sa recherche d'informations. Comme exemples de structures d'accès, on peut citer les menus, les index et le tour guidé. Les structures d'accès possèdent des caractéristiques telles que la définition d'objets cibles et les prédicats sur ces objets (décrivant par exemples les objets étant accessibles).

Les contextes navigationnels permettent de mieux structurer les informations selon la manière choisie par l'utilisateur, ceci afin qu'il ne se perde pas dans l'hyper-espace. Un exemple de contexte navigationnel est donné dans la description de la méthodologie : en effet, si l'on désire parcourir les peintres et les peintures et que nous avons accès à toutes les peintures de Van Gogh, il est souhaitable d'inclure pour chaque peinture quelques informations à son propos.

Cela est différent si nous souhaitons visionner les peintures ayant les fleurs comme sujet (lorsque l'on atteindra les tournesols, il est souhaitable d'avoir quelques informations et références à propos de Van Gogh)[4]. Ainsi, un contexte navigationnel est constitué d'un ensemble de noeuds, de liens, de classes et d'autres contextes navigationnels et est induit de classes navigationnelles telles que les index, les noeuds, les liens et les tour guidés. Il est fait aussi notion de classes de contextes complétant la définition de classes navigationnelles et indiquant quelles informations sont accessibles lorsque l'utilisateur accède à l'objet selon un contexte particulier.

Le schéma navigationnel, quant à lui, représente la navigation envisagée pour l'accès aux informations de l'hypermédia. Ce schéma présente plusieurs caractéristiques afin de mieux percevoir et contrôler la navigation.

D.1.3 Design de l'interface abstraite

Cette étape consiste à définir l'interface qui sera perçue par l'utilisateur et en particulier l'interface de chaque objet navigationnel. Mais c'est ici également que sera définie la manière dont les événements de l'interface seront synchronisés. Un haut degré d'indépendance vis-à-vis de la technologie est atteint du fait de la séparation entre les concepts d'interfaces navigationnelles et d'interface abstraite (différentes interfaces pour le même modèle navigationnel peuvent être construits). L'interface utilisateur d'une application multimédia est décrite selon la « méthode de design de vues abstraites » (ADV). Celles-ci sont en fait des modèles formels d'objets de l'interface définis entre autres par leurs manières de réagir face à des événements externes. Cette méthode n'a d'autres avantages que celui d'offrir un niveau d'abstraction plus élevé face au contexte des hypermédias.

---

[4] ainsi, si la prochaine peinture traite des tournesols il s'agira d'une autre peinture de Van Gogh dans le premier cas, et d'une autre peinture traitant des fleurs dans le second.

On utilise également des diagrammes de configuration afin d'exprimer au mieux la communication entre objets en termes de services fournis ou requis. Ceux-ci sont utilisés dans la méthode ADV afin de représenter les événements extérieurs, le service fourni par la vue abstraite, la communication entre vues et la classe d'objets abstraits. Ces diagrammes permettent d'avoir un aperçu de la façon dont l'utilisateur peut interagir avec l'application hypermédia et en particulier avec les objets définis pour la navigation.

D.1.4 L'implémentation

C'est dans cette étape que le designer doit donner la correspondance entre les objets concrets disponibles dans un environnement donné et le modèle de l'interface abstraite défini par la modélisation de la navigation.

D.2 Scenario-Based Object-Oriented Hypermedi Design Methodology (S.O.H.D.M)

Cette méthodologie est constituée de six phases, à savoir : l'analyse du domaine, la modélisation en tant qu'objet, la modélisation en tant que vue, le design navigationnel, la modélisation pour l'implémentation et la construction. Cette méthodologie est conçue afin de fournir un environnement de bases de données intégré incluant des systèmes hypermédias distribués via l'Internet ou un Intranet. Cette méthodologie se base en outre sur des scénarios afin d'obtenir et d'intégrer les exigences des utilisateurs.

D.2.1 L'analyse du domaine

En premier lieu, les limites du domaine (définissant la frontière du système à développer) sont définies, et des scénario sont créés. En effet, SOHDM formalise les exigences de l'utilisateur à travers la génération de scénario correspondant au travail à effectuer. Afin de faciliter cette génération, la méthode EPC est employée (celle-ci ressemble fortement à un schéma des flux où des indications concernant la navigation ont été joutées). Ce schéma de référence est utilisé dans les phases ultérieures.

D.2.2 La modélisation en tant qu'objet

L'ensemble des scénarios sont utilisés dans cette phase. La création d'objets est réalisée par les phases suivantes :

- l'identification des classes pour chaque scénario : les objets essentiels sont repris, les objets externes et les informations référencées sont candidats pour leur transposition en tant qu'objet. Les objets externes jouent un rôle d'acteur et sont donc des objets dits « actifs ».

- la production de « cartouches » : une alternative à la définition d'objets en tant que données est de les définir en prenant en compte leur rôle dans le modèle.

- la définition des relations : trois types de relations sont à considérer, à savoir les relations de spécialisation, de collaboration et de composition. Le type de relation entrera dans le « cartouche ».

- la vérification du modèle et sa mise à jour:

Le cartouche est ainsi construit, il reprend les informations suivantes: le nom de la classe, des super-classes et des sous-classes, sa responsabilité, ses composants, et sa liste d'attributs.

D.2.3 La modélisation en tant que vue

Dans cette phase, les informations dont on dispose sont réorganisées en tant qu'unités navigationnelles (chaque unité constituant une vue orientée objet). Dans le développement d'applications hypermédias, le principe des vues est, selon cette méthode, hautement recommandé en raison du nombre d'utilisateurs potentiels ayant différentes exigences et également afin de maintenir une plus grande cohérence.

Les vues sont extraites à partir des responsabilités et des attributs repris dans les cartouches. Les attributs peuvent être modifiés et les références croisées sont permises. Les vues seront utilisées comme unités navigationnelles dans la phase suivante.

D.2.4 Le design navigationnel

L'utilisation de scénario et du modèle de donnée améliore la qualité du design navigationnel. De plus, les interfaces utilisateurs sont reprises. Celles-ci sont, entre autre, le Glossaire (racine reliant les concepts similaires), le Menu (organisant la racine comme une table des matières reprenant les concepts majeurs), le Menu hiérarchique et enfin la Page de recherche.

La première tâche à effectuer dans cette phase est de déterminer les interfaces qui seront employées, ensuite, les liens navigationnels (une vue source, une vue cible ainsi que des attributs définissant le lien) sont définis. A cet effet, une matrice caractérisant les types de liens entre les vues peut être utilisée.

D.2.5 La phase d'implémentation

Cette phase génère la structure des pages, leur flux et l'interface utilisateur. Elle est voulue transparente vis-à-vis des environnements pouvant être utilisés, des règles peuvent être employées afin de faciliter les transformations pour une base de données ou tout autre type de système dérivé. Finalement, la structure des pages est peaufinée afin d'y inclure la localisation des données, les propriétés des composants et le choix des interfaces.

D.3 The Enhanced O-R Model (E.O.R.M)

Cette méthodologie propose une méthode orientée objet basée sur un processus de modélisation capturant la structure, le comportement et l'interaction des objets d'un domaine d'application. Elle permet entre autres de capturer de façon adéquate la sémantique complexe des interactions entre objets caractérisant les applications hypermédias.

D.3.1 Le modèle orienté objet

Les objets sont utilisés afin de modéliser les entités du monde réel et de connaître leur identité. Deux mécanismes de structuration caractérisent cette approche orientée objet. Le premier, appelé classification, constitue une catégorisation d'objets; les objets ayant une structure commune et une sémantique équivalente sont rassemblés dans une classe afin de modéliser des ensembles homogènes d'entités. Le second mécanisme, appelé généralisation, représente les situations où les classes sont combinées afin de former de nouvelles classes définissant un ensemble de sémantiques structurelles et de comportements communs et partagés[5].

D.3.2 Amélioration du modèle orienté objet

Un modèle orienté objet représente une relation binaire et bidirectionnelle entre deux objets en définissant un attribut pour chaque objet pointant vers un autre. Mais une telle représentation ne donne pas la sémantique des relations.

Cette méthodologie emploie un autre type de schéma, un modèle orienté objet basé sur l'intégration du modèle orienté objet habituel et de la sémantique orientée objet (contraintes sur les cardinalités, attributs de la relation et restrictions sur les relations). Ceci permet, entre autres, de représenter plus facilement l'environnement de l'application hypermédia. L'idée centrale est de représenter la sémantique des relations dans le modèle orienté objet. La sémantique concernant les relations est définie comme une classe permettant à la sémantique comportementale et structurelle d'être représentées en un seul endroit. Les structures hypermédias peuvent aisément être représentées à l'aide du schéma OO amélioré.

---

[5] Par exemple, la classe "voiture" et "avion" peut être combinée afin de former une nouvelle classe "véhicule". Chaque objet "voiture" sera donc indirectement un membre de la classe "véhicule" et héritera de ses propriétés.

D.3.3 La méthode de design OO

Cette méthode, basée sur le nouveau schéma OO, permet de capturer la sémantique concernant l'interaction entre objets dans un système orienté objet. Elle est constituée de trois composantes principales : « The class framework », « The composition framework » et enfin le « GUI framework ».

D.3.3.1 « The Class Framework »

Cette étape consiste en une librairie de définitions de classes.

- L'identification des classes

La première étape consiste à identifier les classes du domaine d'application. Ce processus est essentiellement basé sur des critères personnels (un exemple de classe pourrait être constitué d'utilisateurs, d'une date sensée être reprise dans un mail, d'un fichier ou système de fichiers). Il faut toutefois faire attention à ne garder que les classes pertinentes ainsi que les classes pouvant être correctement définies et dont on a une idée claire. De plus, les classes n'ayant pas une existence indépendante sont considérées comme attributs d'autre classes.

- Le raffinement des classes

Les classes sont détaillées et des attributs, des opérations ainsi que des relations d'héritage sont ajoutés. Seuls les attributs pertinents ne pouvant être dérivés sont ajoutés ; ceux-ci sont vus comme des propriétés des classes. Chaque objet possède une identité unique et si un attribut peut avoir un cycle de vie indépendant, il est considéré comme un objet et non comme un attribut.

- L'ajout d'opérations

Les opérations définissent le comportement des classes ; elles permettent aux designers d'encapsuler leurs connaissances à propos d'une certaine classe.

- L'héritage

Il existe des situations dans lesquelles certaines classes distinctes ont besoin d'être regroupées ; cela peut être perçu comme une généralisation des sémantiques communes en une nouvelle super-classe ou comme une spécialisation des nouvelles classes existantes en de nouvelles sous-classes.

D.3.3.2 « The composition Framework »

Cette étape consiste en une librairie de définitions de classe de liens. Deux activités sont définies : l'identification des compositions et leur raffinement.

- L'identification des compositions

Une composition est constituée d'une relation, des classes participant à la relation, de leurs cardinalités et de la sémantique de la relation. C'est donc dans cette sous-phase que sont identifiées les compositions des classes à l'intérieur du domaine d'application. Il s'agit encore de ne prendre en compte que les informations pertinentes et atomiques. On obtient alors une bonne idée des compositions du domaine de l'application.

- Le raffinement des compositions

Ce raffinement est effectué notamment en définissant le domaine et les cardinalités des « participants », ainsi que des invariants et une classe de liens sémantiques pour chaque composition. Les invariants peuvent être répartis en deux catégories. La première concerne les invariants liés aux liens individuels et la deuxième, les invariants des compositions.

Quant à la définition de la classe de liens sémantiques, nous devons, pour chaque relation dans chaque composition, considérer quelques problèmes, à savoir la prise en compte de la direction du lien, de la fréquence d'utilisation, d'une liaison davantage dynamique ou plutôt statique,...

D.3.3.3 « The GUI Framework »

Cette étape consiste en une librairie réutilisable de définitions et de présentations ; deux activités y sont définies : l'identification des fenêtres et de la présentation et le transfert des classes et des compositions en présentation. Cette partie n'est pas considérée comme importante dans EORM et elle est très brièvement définie.

D.3.4 L'évaluation

Dans cette méthodologie, le design orienté objet est perçu comme un processus d'itération. A chaque étape, il est possible de procéder à un retour en arrière afin de corriger ou de raffiner la modélisation. En pratique, la modélisation et la construction d'un système d'information hypermédia ne sont pas aussi ordonnés et souvent, il est possible de construire un prototype de l'interface utilisateur bien avant le dernier moment.

Annexe E. Code source du programme de l'étude de cas

E.1 index.html

```
<HTML>
<SCRIPT>
function WinOpener() {

msgWindow=window.open("","displayWindow","menubar=no,scrollbars=no,status
=yes,width=700 ,height=640 ,resizable=yes")
   msgWindow.document.write  ("<HEAD><TITLE>Test avec Heitml et les
lunettes</TITLE></HEAD>")
   msgWindow.document.write  ('<FRAMESET COLS="520,*"><FRAMESET
ROWS="540,*"><FRAME SRC="start.html" name="lunettes"><FRAME
SRC="frame_bas.html" name="info"></FRAMESET><FRAME SRC="frame_droit.html"
name="menu_droit"></FRAMESET>')
   msgWindow.document.write  ("<BODY TARGET='menu'>")
}
</SCRIPT>
<HEAD><TITLE>Test avec Heitml et les lunettes</TITLE></HEAD>
<BODY bgcolor="#ffffff" onLoad=WinOpener()>
<H1>Test avec Heitml et les lunettes</H1>
<HR>
</BODY>
</HTML>
```

E.2 start.html

```
<HTML>

<HEAD><TITLE>Test avec Heitml et les lunettes</TITLE></HEAD>

<BODY bgcolor="#ffffff">
<U><H2>Inscription</H2></U>
<H4>Si vous possédez un mot de passe, n'entrer que celui-ci et votre
nom.</H4>
<FORM ACTION=start.hei>
<TABLE BORDER bgcolor="#f5f5dc">
<TR bgcolor="#fff0dc">
      <TD> Nom : </TD>
      <TD> <input name="nom" size=25> </TD>
</TR>

<TR>
      <TD> Prénom :  </TD>
      <TD> <input name="prenom" size=25> </TD>
</TR>

<TR bgcolor="#fff0dc">
      <TD> Mot de passe :  </TD>
      <TD> <input type=password name="password" size=25> </TD>
</TR>

<TR>
      <TD> Sexe :  </TD>
      <TD> <select name="sexe"><option value="H" selected>Homme<option
value="F">Femme</select> </TD>
</TR>

<TR>
      <TD> Age :  </TD>
      <TD> <input name="age" size=2> </TD>
```

```
</TR>

<TR>
      <TD> Rue :  </TD>
      <TD> <input name="rue" size=25> </TD>
</TR>

<TR>
      <TD> Code Postal :  </TD>
      <TD> <input name="code_postal" size=25> </TD>
</TR>

<TR>
      <TD> Localité :  </TD>
      <TD> <input name="localite" size=25> </TD>
</TR>

<TR>
      <TD> Pays :  </TD>
      <TD> <input name="pays" size=25> </TD>
</TR>

<TR>
      <TD> Num Carte Crédit :  </TD>
      <TD> <input name="num_cc" size=25> </TD>
</TR>

<TR>
      <TD> Email :  </TD>
      <TD> <input name="email" size=25> </TD>
</TR>

</TABLE>
<br>
<INPUT TYPE="submit" VALUE="S'inscrire">
</FORM>
</BODY>
</HTML>
```

E.3 start.hei

```
<include ses.hei>
<include datetime.hei>
<HTML>
<HEAD>
<TITLE>

</TITLE>
</HEAD>
<BODY bgcolor="#ffffff">

<let m="std"; srvmax =1;
if !isempty(ff.nom) && !isempty(ff.password)>
      <dbquery q >SELECT * FROM clients WHERE nom = <? ff.nom quoted> AND
password = <? ff.password quoted>
      <dbrow>
            <let m="create";
            let se.ident_client = q.ident;
            let se.nom = q.nom;
            let se.password = q.password;
            let se.prenom = q.prenom;
            let se.age = q.age;
            let se.sexe = q.sexe;
```

```
            let se.rue = q.rue;
            let se.code_postal = q.code_postal;
            let se.localite = q.localite;
            let se.pays = q.pays;
            let se.num_cc = q.num_cc;
            let se.email = q.email;>
            <H1>Bienvenue <? se.prenom> <? se.nom></H1><br>
            <H3>Vos précédents achats sont:</H3>
            <let srvmax = null>
            <dbquery r>SELECT * FROM achats WHERE ident_client = <?
se.ident_client>
                  <dbrow>
                  <dbquery p> select * from lunettes where ident = <?
r.ident_lunettes>
                        <dbrow><img src="images/<? p.fichier_face>"><br>
                              <TABLE BORDER bgcolor="#f5f5dc">
                              <TR>
                                    <TD> Numéro du modèle : </TD>
                                    <TD> <? p.num_cat> </TD>
                              </TR>

                              <TR>
                                    <TD> Couleur des verres :  </TD>
                                    <TD> <? p.couleur_verres> </TD>
                              </TR>

                              <TR>
                                    <TD> Couleur de la monture :  </TD>
                                    <TD> <? p.couleur_monture></TD>
                              </TR>

                              <TR>
                                    <TD> Materiau :  </TD>
                                    <TD> <? p.materiau></TD>
                              </TR>

                              <TR>
                                    <TD> Type :  </TD>
                                    <TD> <? p.type></TD>
                              </TR>

                              </TABLE>

                  </dbquery>
                  <dbempty>Auncuns
            </dbquery>
      <dbempty>
            <let m="failed">
            Mot de passe incorrecte ou utilisateur inexistant
            <a href="start.html">Retour</a>

      </dbquery>
<else>
<if contains(default(ff.email),"@") && !isempty(ff.nom) &&
!isempty(ff.prenom) && !isempty(ff.sexe) && !isempty(ff.age) &&
!isempty(ff.rue)
&& !isempty(ff.code_postal) && !isempty(ff.localite) && !isempty(ff.pays)
&& !isempty(ff.num_cc);

      let m="create";
      let se.nom = ff.nom;
      let se.prenom = ff.prenom;
      let se.age = ff.age;
      let se.sexe = ff.sexe;
      let se.rue = ff.rue;
      let se.code_postal = ff.code_postal;
```

```
      let se.localite = ff.localite;
      let se.pays = ff.pays;
      let se.num_cc = ff.num_cc;
      let se.email = ff.email;>
      <let se.password = ff.nom + ff.age>
      <dbupdate>
            INSERT INTO clients
            (nom, password, prenom, age, sexe, rue, code_postal,
localite, pays, num_cc, email)
            VALUES
            (<? se.nom quoted>,
            <? se.password quoted>,
            <? se.prenom quoted>,
            <? se.age quoted>,
            <? se.sexe quoted>,
            <? se.rue quoted>,
            <? se.code_postal quoted>,
            <? se.localite quoted>,
            <? se.pays quoted>,
            <? se.num_cc quoted>,
            <? se.email quoted>)
      </dbupdate>
      <let srvmax = null>
      <dbquery q> select ident from clients
            <dbrow> <let se.ident_client = q.ident>
      </dbquery>
      <H1>Bienvenue <? se.prenom> <? se.nom></H1><br>
      <H4>
      Votre mot de passe est :<TABLE BORDER bgcolor="#f5f5dc"><TR><TD><?
se.password></TD></TR></Table><BR>


   <else

      let m="failed";
      >Données incorrectes ou manquantes
      <a href="start.html">Retour</a>
      <
/if;
/if;
>

<if m != "failed">
<session "create">
<H4>Vous pouvez passer à la phase de sélection en cliquant sur ce <a
href="lunettes.hei?<sessionurl>&age=<? se.age>&sexe=<?
se.sexe>">lien</a>.
<br>Ensuite cliquez sur les zones de la lunette</H4>

</session>

</if>
</BODY>
</HTML>
```

E.4 lunettes.hei

```
<include ses.hei>
<include htmlext.hei>

<HTML>
<BODY bgcolor="#ffffff" background="images/pal2.jpg"><base
target="menu_droit">
```

```
<session>

<if !isempty(ff.age)>
      <if (se.age) <= 30> <let se.type = 'moderne'>
        <else> <if (se.age) <=55> <let se.type = 'classique'>
          <else> <let se.type='tradition'>
      </if></if>

      <let srvmax=1>
      <dbquery q> SELECT * FROM lunettes WHERE sexe like '%<? se.sexe>%'
AND type=<? se.type quoted>
            <dbrow>
            <br><br><br><br><br><br><br><br><br>
            <center><IMG SRC="images/<? q.fichier_face>"
USEMAP="#lunettesmap1" border=0 ></center>
            <MAP NAME="lunettesmap1">
                  <AREA SHAPE="rect" COORDS="0,15,95,75"
HREF='menu_droit.hei?<sessionurl>&choix=coulv'
OnMouseOver="self.status='Changer la couleur des verres';return true">
                  <AREA SHAPE="rect" COORDS="95,0,125,45"
HREF='menu_droit.hei?<sessionurl>&choix=modele'
OnMouseOver="self.status='Changer le modèle de la monture'; return true">
                  <AREA SHAPE="rect" COORDS="125,0,220,75"
HREF='menu_droit.hei?<sessionurl>&choix=coulv'
OnMouseOver="self.status='Changer la couleur des verres';return true">
            </map>
            <let se.ident_lunettes = q.ident>
            <let se.num_cat = q.num_cat>
            <let se.coulv = q.couleur_verres>
            <let se.coulm = q.couleur_monture>
            <let se.materiau = q.materiau>
            <let se.type = q.type>
      </dbquery>
      <br><br><br><br><br><br><br><br><br>
      <script>
            top.info.location="info.hei?<sessionurl>"
      </script>

<else>
      <let srvmax=1>

      <if ff.choix == "coulv">
      <let se.coulv = ff.coulv>
      <else> <if ff.choix == "coulm">
            <let se.coulm = ff.coulm>
            <else> <if ff.choix == "type">
                  <let se.type = ff.type>
                  <else> <if ff.choix == "modele">
                        <let se.num_cat = ff.num_cat>
                        </if>
                  </if>
            </if>
      </if>

      <let se.ident_lunettes = ff.ident>
      <dbquery q> SELECT * FROM lunettes WHERE num_cat = <? se.num_cat
quoted> AND couleur_verres = <? se.coulv quoted>
            <dbrow>
            <br><br><br><br><br><br><br><br><br>
            <center><IMG SRC="images/<? q.fichier_face>"
USEMAP="#lunettesmap1" border=0 ></center>
            <MAP NAME="lunettesmap1">
                  <AREA SHAPE="rect" COORDS="0,15,95,75"
HREF='menu_droit.hei?<sessionurl>&choix=coulv'
OnMouseOver="self.status='Changer la couleur des verres';return true">
```

```
                  <AREA SHAPE="rect" COORDS="95,0,125,45"
HREF='menu_droit.hei?<sessionurl>&choix=modele'
OnMouseOver="self.status='Changer le modèle de la monture'; return true">
                  <AREA SHAPE="rect" COORDS="125,0,220,75"
HREF='menu_droit.hei?<sessionurl>&choix=coulv'
OnMouseOver="self.status='Changer la couleur des verres';return true">
            </map>
      </dbquery>
      <br><br><br><br><br><br><br><br><br><br>
      <script>
            top.info.location="info.hei?<sessionurl>"
      </script>


</if>
<center><table border bgcolor="#f5f5dc"><TR><TD><a
href="command.hei?<sessionurl>" target =
_parent>Commander</a></TD></TR></TABLE></center>

</session>
</center>

</BODY>
</HTML>
```

E.5 info.hei

```
<include ses.hei>
<HTML>
<BODY>
<session>
      <TABLE BORDER bgcolor="#f5f5dc">
            <TR>
                  <TD> Num modèle : </TD>
                  <TD> Coul. monture :  </TD>
                  <TD> Coul. verres :  </TD>
                  <TD> Materiau :  </TD>
                  <TD> Type :  </TD>
            </TR>

            <TR>
                  <TD> <? se.num_cat> </TD>
                  <TD> <? se.coulm></TD>
                  <TD> <? se.coulv> </TD>
                  <TD> <? se.materiau></TD>
                  <TD> <? se.type></TD>
            </TR>

      </TABLE>

</session>
</BODY>
</HTML>
```

E.6 menu_droit.hei

```
<include ses.hei>
<HTML>
<BODY bgcolor="#ffffff">
<session>
<center>
```

```
<base target = 'lunettes'>


<if (ff.choix) == "coulv">
<H3>Couleurs des verres</H3>
<dbquery q> SELECT * FROM lunettes WHERE num_cat = <? se.num_cat quoted>
AND not(couleur_verres=<? se.coulv quoted>)
<dbrow>
<a href="lunettes.hei?<sessionurl>&ident=<? q.ident>&choix=coulv&coulv=<?
q.couleur_verres>"> <img src="images/<? q.fichier_face>" , width=110
,height=35, border =0></a>
<br>
<? q.couleur_verres>
<br><br>
<dbempty>
Il n'y a pas de montures ayant des verres de couleur differente ayant les
memes caracteristiques.
</dbquery>
</if>

<if (ff.choix) == "modele">
<H3>Modèle de lunettes</H3>
<dbquery q> SELECT * FROM lunettes WHERE not(num_cat = <? se.num_cat
quoted>) AND couleur_verres=<? se.coulv quoted>
<dbrow>
<a href="lunettes.hei?<sessionurl>&ident=<?
q.ident>&choix=modele&num_cat=<? q.num_cat>"> <img src="images/<?
q.fichier_face>" , width=110 ,height=35 border=0></a>
<br>
<? q.num_cat>
<br><br>
<dbempty>
Il n'y a pas de montures de modèle different ayant les memes
caracteristiques.
</dbquery>
</if>

</center>
</session>
</BODY>
</HTML>
```